Statistical Inference of Diffusion Networks

Huang, Hao; Yan, Qian; Chen, Lu; Gao, Yunjun; Jensen, Christian S.

Published in:
I E E E Transactions on Knowledge & Data Engineering

DOI (link to publication from Publisher):
10.1109/TKDE.2019.2930060

Publication date:
2021

Document Version
Accepted author manuscript, peer reviewed version

Response Letter for TKDE-2019-03-0210

We thank the reviewers for taking time in helping us to improve this manuscript. We appreciate the constructive comments, and we have tried our best to address them in this new version. Below, we first summarize minor revisions, and then, we provide detailed responses to all the comments.

List of the minor revisions

1. In the second paragraph of Section 4.4 (Algorithm), we have re-written the explanation of algorithm steps with more details.
2. At the end of Section 5.2 (Effect of Diffusion Network Size), we have added a paragraph to further discuss the effect of network size to the running time and space requirement of tested algorithms, which also explains the reason why we use current network sizes.
3. In Section 6 (Conclusion), we have added a paragraph to explain why the SIDM algorithm would fail to recover a dynamic diffusion network whose node set or edge set changes over time.
4. We have carefully proofread this manuscript multiple times in order to further improve its presentation.

As a result, we believe that we have satisfactorily addressed all the reviewers' concerns within the space constraint.

Detailed response

Response to reviewer #1:

Comment 1.1: *Authors have taken a very good problem and come up with some good result that will advance the state of the arts. I have a few concerns about the manuscript. (1) The size of the network taken for experimental results is very small. It should be at least 10000 nodes.*

Response 1.1: Thanks for the encouraging comments and valuable suggestion. In fact, based on extensive testing, we have found that the running time of the NetRate algorithm and MulTree algorithm rapidly increase with the growth of network size, and exceeds acceptable levels even with a medium network size of 1000. On the other hand, when the network size exceeds 1000, execution of the SIDN algorithm often witnesses out-of-memory errors. The reason is that NetRate and MulTree consider all propagation paths supported by each cascade without any pruning strategy, while SIDN records the MI (Mutual Information) value for each node pair for its MI-based pruning method. Thus, if the objective network contains more nodes, there will be significantly more possible propagation paths and relatively more node pairs, incurring higher computational costs for NetRate and MulTree and a larger space requirement for SIDN. Therefore, we have selected and tested networks with sizes varying from 100 to 750. To help our readers better understand the effect of network size to the tested algorithms as well as the reason why we use such network sizes, we provide the above explanation in Section 5.2 (Effect of Diffusion Network Size).

Comment 1.2: *(2) The explanation of steps is generally missing and that requires effort in understanding particularly where intermediate steps have been missed.*

Response 1.2: Thanks for pointing this out. We have revised the explanation of algorithm steps in the second paragraph

of Section 4.4 (Algorithm), with more details and hopefully better clarity. Our goal is that the revision will be helpful for our readers to better understand the SIDN algorithm.

Comment 1.3: *(3) Language needs some improvement and it should be checked again.*

Response 1.3: Thanks for the valuable suggestion. We have tried our best by requesting the available proof-reading service to correct grammatical errors and typos and improve the English presentation of the paper. We hope the presentation of this revision be logically smooth and grammatically acceptable without affecting the readers to get the main points.

Response to reviewer #2:

General Comment 2.1: *An algorithm (SIDN) is provided to infer diffusion network structure based only the final infection status, not demanding monitoring of infection timestamps as diffusion process occurs, nor prior knowledge of the network. This leads to interesting applications to field s such as epidemic modelling and prevention an viral marketing. Accuracy and complexity of the algorithm are satisfactory. Paper is well-written, clearly explains its goals, interesting and well written paper, deserves publication.*

Response 2.1: Thanks for all the encouraging comments from the reviewer.

Response to reviewer #3:

General Comment 3.1: *The method learns diffusion network structures based only on the final infection statuses of nodes. This approach does not rely on monitoring the infection timestamps of nodes as a diffusion process occurs. I have only one simple query. If the nodes enter or leave the network or the set of potential parents changes over time, is the network resilient to this? What would be the effect of changes if the infection timestamps are not observed? I think, author should discuss on the topic as they are not considering the infection timestamp.*

Response 3.1: Thanks for the encouraging comments and valuable suggestion. Our SIDN algorithm is proposed to infer influence relationships (i.e., edges) in diffusion networks with static structures. Thus, it focuses on the final statuses of nodes at the end of each diffusion process. If the node set or edge set changes over time, SIDN will miss many intermediate influence relationships and thus fail to recover a dynamic diffusion network. In this new version, we have pointed this out at the end of Section 6 (Conclusion), as suggested.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2019.2930060, IEEE Transactions on Knowledge and Data Engineering

Statistical Inference of Diffusion Networks

Hao Huang, Qian Yan, Lu Chen, Yunjun Gao, *Member, IEEE,* and Christian S. Jensen, *Fellow, IEEE*

Abstract—To infer structures in diffusion networks, existing approaches mostly need to know not only the final infection statuses of network nodes, but also the exact times when infections occur. In contrast, in many real-world settings, such as disease propagation, monitoring exact infection times is often infeasible due to a high cost. We investigate the problem of how to learn diffusion network structures based on only the final infection statuses of nodes. Instead of utilizing sequences of timestamps to determine potential parent-child influence relationships between nodes, we propose to find influence relationships with high statistical significance. To this end, we design a probabilistic generative model of the final infection statuses to quantitatively measure the likelihood of potential structures of the objective diffusion network, taking into account network complexity. Based on this model, we can infer an appropriate number of most probable parent nodes for each node in the network. Furthermore, to reduce redundant inference computations, we are able to preclude insignificant candidate parent nodes from being considered during inferencing, if their infections have little correlation with the infections of the corresponding child nodes. Extensive experiments on both synthetic and real-world networks offer evidence that the proposed approach is effective and efficient.

Index Terms—Diffusion network, influence relationship, infection timestamp, probabilistic generative model

1 Introduction

A Diffusion network is a directed graph where an edge from a parent to a child indicates that the parent influences the child. Diffusion network inference aims to reveal unknown influence relationships between nodes in a diffusion network based on observed diffusion results. This problem has received considerable attention in recent years, in areas such as social networks [6], information propagation [13], epidemic prevention [30], and viral marketing [18]. Inferred diffusion network structures enable an intuitive understanding of the underlying interactions between nodes, and they help better predict, promote, or prevent future diffusions.

Existing approaches to diffusion network inference mostly assume that diffusion results are available that consist of both the final infection statuses of nodes and the exact times when infections occurred. In these approaches, nodes infected sequentially within a time interval are assumed to posses influence relationships, the previously infected ones being regarded as potential parents of the subsequently infected ones [20].

Exact infection timestamps of nodes are sometimes straightforwardly available, for example, in online social networks. However, in many real-world diffusion processes, such timestamps are often unavailable. Examples include the spread of epidemics and the viral marketing campaigns [2]. Conducting a comprehensive survey of the prevailing situation and marketing results are time-consuming and expensive and cannot be conducted frequently. Even if frequent comprehensive surveying is affordable, due to the different incubation periods (i.e., the time from infection to outbreak) and the uncertainties of respondents' feedbacks, the obtained temporal information is unlikely to reflect the exact occurrence times of infections.

In order to conduct diffusion network inference without infection times, new techniques are required that are able to learn parent-child influence relationships, as it is no longer possible to directly attribute the infection of a node to particular previously infected parent nodes. To the authors' knowledge, only two existing studies [2], [12] have partially addressed the problem of inferring diffusion networks without infection timestamps of nodes. These studies require the availability of either all triples that capture the nodes that are connected in diffusion paths (although diffusion paths are not naturally visible or traceable in many diffusion processes) or prior knowledge on the number of directed edges in the objective diffusion network, which is also hard to obtain in practice.

In contrast, we investigate the problem of learning diffusion network structures from only the final infection statuses of the nodes in historical diffusion processes. We propose an effective and efficient approach called SIDN (**S**tatistical **I**nference of **D**iffusion **N**etworks) to solving this problem. SIDN reveals statistically significant influence relationships by finding for each node a set of potential parent nodes that are most likely to have generated the observed final infection statuses. To achieve this, we develop a probabilistic generative model based on relative entropy (*a.k.a.* Kullback-Leibler divergence) to quantify the fit between an inferred diffusion network structure and the observed final infection statuses. For each network node, the candidate parent nodes bringing lower relative entropy are considered to influence the node with higher probability. In addition, we also take into account network complexity during parent node selection, and we are able to derive a theoretical upper bound on parent node set size, which in turn helps SIDN to control the complexity of the inferred diffusion network structure, thus avoiding too many inferred low-

- *H. Huang and Q. Yan are with the School of Computer Science, Wuhan University, Wuhan 430072, China. E-mail: {haohuang, qy}@whu.edu.cn.*
- *L. Chen and C. S. Jensen are with the Department of Computer Science, Aalborg University, Aalborg DK-9220, Denmark. E-mail: {luchen, csj}@cs.aau.dk.*
- *Y. Gao (corresponding author) is with the College of Computer Science and Technology, Zhejiang University, Hangzhou 310027, China. E-mail: gaoyj@zju.edu.cn.*

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2019.2930060, IEEE Transactions on Knowledge and Data Engineering

probability influence relationships that do not exist in the corresponding real diffusion network. Furthermore, in order to reduce redundant computations during the structure inferencing, SIDN is able to identify candidate parent nodes that are statistically insignificant by checking whether their infections are independent or have extremely low correlation with the infections of the corresponding child nodes. Such insignificant candidates can then be disregarded from consideration as potential parent nodes.

In summary, our key contributions include the following: (1) Departing from existing approaches that rely on infection timestamps, we propose a new statistical approach that can infer diffusion network structure based on only the final infection statuses of nodes, which are much more easily available in practice. This approach does not rely on monitoring the infection timestamps of nodes as a diffusion process occurs, and correct observed infection timestamps are not needed. The approach only needs the final infection statuses of nodes; other information, including prior knowledge of the network, is not needed. (2) Our approach takes into account both of the accuracy and complexity of the inference result to find an appropriate number of parent nodes for each network node, thus avoiding overly complex inferred structures that are conceptually and computationally intractable. (3) We present a heuristic pruning method for the screening of candidate parent nodes that enables our approach to eliminate redundant computations.

The remainder of the paper is organized as follows. We review the related work in Section 2 and present a problem statement in Section 3. We then cover the proposed SIDN algorithm in Section 4, and we report experimental findings in Section 5 and then offer conclusions in Section 6.

2 Related Work

The existing approaches to diffusion network inference can be categorized into two main groups: (1) infection timestamp-based approaches, and (2) infection timestamp-free approaches.

2.1 Infection Timestamp-Based Approaches

Most existing approaches to diffusion network inference require the temporal information of node infections, assuming that the observed diffusion results used by them (known as cascades) record the exact infection timestamp of each infected node in every diffusion process. Three main types of infection timestamp-based approaches have been proposed: (1) the convex programming-based approaches, (2) the submodularity-based approaches, and (3) the embedding-based approaches.

Convex programming-based approaches try to find diffusion network structures that maximize the likelihood of given cascades based on convex optimization. To approximate optimal solutions, these approaches utilize different techniques, such as sequential quadratic programming [7], [21], the EM algorithm [27], [31], block coordinate descent [5], stochastic and proximal gradient methods [4], [10], survival theory [9], sparse recovery [26], and decoupling into multiple parallelizable problems [15], [22], [23], to solve their optimization problems. These approaches generally exhibit nice inference performance on tree-like or sparse networks.

Submodularity-based approaches transform the problem of diffusion network inference into a problem of submodular optimization, as they use likelihood functions of cascades for given propagation trees that have the property of submodularity. NetInf [8] and MulTree [11] are state-of-the-art approaches of this type. Due to the submodularity of their objective functions, both approaches adopt a greedy algorithm to achieve a near-optimal solution. The main difference between them is that during the submodular optimization, NetInf considers only the most probable propagation tree, to achieve high efficiency, while MulTree considers all propagation trees supported by each cascade, to achieve high accuracy.

Embedding-based approaches map the nodes in observed diffusion process into a latent embedding space, in which the distance between each two mapped nodes represents the transmission rate (or propagation probability). These approaches model the transmission rates using Weibull distributions [16], uniform distributions [6], or via kernels [3], and they learn the transmission rates between nodes based on observed cascades. Although embedding-based approaches do not explicitly reveal the diffusion network structures, they enable users to observe influence relationships between nodes via low-dimensional spaced visualizations.

The above three types of paradigms for diffusion network inference all require complete and correct cascades. Abrahao et al. [1] have proven that with an adequate amount of complete and correct cascades, the objective diffusion network can be inferred accurately using simple reconstruction approaches. Nevertheless, in reality, observed cascades may have partially incorrect infection timestamps, and they may miss partial snapshots of the network. Several methods have been proposed to mitigate the effects of partially incorrect [28] or missing infection timestamps [14], [19]. These methods are complementary to the above three types of approaches.

Departing from the infection timestamp-based approaches, our SIDN algorithm requires only the final infection statuses of nodes, which are much more easily accessible in many real-world diffusion processes. Therefore, SIDN has a wider range of applicability, and is also unaffected by incorrect and missing timestamps.

2.2 Infection Timestamp-Free Approaches

So far, two approaches have been proposed to infer diffusion network structures without the help of node infection timestamps. Both approaches learn the influence relationships between nodes, either from path traces (referred to as the PATH approach) or based on lifting effects (referred to as the LIFT approach).

PATH takes as inputs path-connected triples, each of which is a set of three nodes that are activated along a diffusion path through a network. It inserts edges between the nodes that co-occur most frequently in the path-connected triples [12]. This approach has nice properties such as a solid mathematical foundation and low computational cost. However, it requires path-connected triples, which are often difficult or impossible to obtain from natural diffusion processes. Even if complete and correct cascades are available, inferring exact path-connected triples is still difficult.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2019.2930060, IEEE Transactions on Knowledge and Data Engineering

LIFT studies the problem of diffusion network inference in the case that only initial and final infection statuses of nodes are available [2]. It calculates the lifting effect of each node u to another node v, which measures the increase in the probability of v's infection on the condition that u is previously infected. LIFT discovers an edge by finding two nodes with the currently largest lifting effect. Without prior knowledge on the number of edges in the network, this approach will iteratively add the discovered edges until all nodes are connected.

Compared with the above two infection timestamp-free approaches, the SIDN algorithm only requires the final infection statuses of network nodes and does not rely on any other information on infections or on prior knowledge on the network. Therefore, the SIDN algorithm is more widely applicable in practice.

3 Problem Statement

A diffusion network is 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 nodes. An 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 infect v_j with a certain probability (which can be regarded as the weight of the edge). Table 1 lists notation that will be used henceforth.

As a few existing studies offer proposal for how to calculate edge weights based on observed infection status results [32]. In contrast, we focus on inferring the unknown directed edge set of the objective network. Formally, our problem statement 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 historical diffusion processes, where $S^\ell = (x_1^\ell, ..., x_n^\ell)$ is a n-dimensional vector that records the final infection status, $x_i^\ell \in \{0, 1\}$ (0 denotes uninfected, and 1 denotes infected) of each node $v_i \in V$ observed at the end of the ℓ-th diffusion process ($\ell \in \{1, \ldots, \beta\}$).

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

In the problem statement, except for the given infection status results S observed on the n nodes of the objective diffusion network G, no other information about infections and the network, such as infection timestamps, initially infected nodes, and the number m of directed edges in the network, is known.

4 The SIDN Algorithm

We first explain how to measure the likelihood of inferred diffusion network structures by presenting a relative entropy-based probabilistic generative model of the node infection statuses, followed by explaining how to trade off relative entropy versus network complexity during diffusion network inference. Then we present how to prune statistically insignificant candidate parent nodes to eliminates redundant computations before giving the detailed steps of the SIDN algorithm. We finally offer a complexity analysis on the SIDN algorithm.

TABLE 1
Notation

Symbol	Description
G	A directed graph.
V	The set of nodes in G.
n	The number of nodes in G.
v_i	The i-th node in V ($1 \leqslant i \leqslant n$).
E	The set of directed edges in G.
m	The number of directed edges in G.
S	The infection statuses of nodes in G observed across β diffusion processes.
α	The initial infection ratio of nodes.
β	The number of diffusion processes on G.
x_i^ℓ	The infection status of node v_i in the ℓ-th diffusion process ($1 \leqslant i \leqslant n, 1 \leqslant \ell \leqslant \beta$).
π_i^ℓ	The infection statuses of v_i's parent nodes in the ℓ-th diffusion process ($1 \leqslant i \leqslant n, 1 \leqslant \ell \leqslant \beta$).
X_i	The infection status variable of node $v_i \in V$.
F_i	The parent node set of node v_i.
$\|F_i\|$	The number of nodes in F_i.
X_{F_i}	The set of infection status variables of nodes in F_i.
$p^*(X_1, \ldots, X_n)$	The true joint probability distribution of node infection statuses.
$p(X_1, \ldots, X_n)$	The joint probability distribution of node infection statuses calculated by S.
$KL(p^*, p)$	The relative entropy between $p^*(X_1, \ldots, X_n)$ and $p(X_1, \ldots, X_n)$.
$H(X_i, \ldots, X_n)$	The entropy of variables $\{X_1, \ldots, X_n\}$.
$H(X_i \| X_{F_i})$	The entropy of X_i conditioned on X_{F_i}.
$p(x, f, w)$	The joint probability that random variable X takes value x and $X_{F,W}$ takes values f and w.
$g(v_i, F_i)$	The selection criterion for parent node set F_i of node v_i.
λ	The coefficient of the penalty term in the scoring function of parent node sets.
$MI(X_i, X_j)$	The mutual information between the infection statuses of nodes v_i and v_j.
τ	A threshold for mutual information.
P_i	The set of candidate parent nodes of node v_i ($\forall v_j \in P_i, MI(X_i, X_j) \geqslant \tau$).
C_i	The set of possible combinations of v_i's candidate parent nodes.

4.1 Probabilistic Generative Model

In a diffusion network, since nodes are infected by other nodes via the directed edges of the network, the diffusion result depends on the diffusion network structure. Therefore, diffusion network inference is equivalent to finding a diffusion network structure that is most likely to have generated an observed diffusion result. In order to carry out diffusion network inference with only the node infection status results S observed across β diffusion processes, the essential work is to infer the probabilistic generative model of S, i.e., the true joint probability distribution $p^*(X_1, ..., X_n)$ over the variables $X_1, ..., X_n$ of node infection statuses. Moreover, since the infection of a node can be affected only by its parent nodes during the diffusion processes, we can reformulate

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2019.2930060, IEEE Transactions on Knowledge and Data Engineering

$p^*(X_1, ..., X_n)$ as follows.

$$p^*(X_1, ..., X_n) = \prod_{i=1}^{n} p^*(X_i \mid X_{F_i^*}), \tag{1}$$

where F_i^* is the set of true parent nodes of v_i and $X_{F_i^*}$ is the variables of the infection statuses of v_i's true parent nodes.

Then, our goal is to find a diffusion network G so that its corresponding probability distribution $\prod_{i=1}^{n} p(X_i \mid X_{F_i})$ is an as good as possible approximation of approximates as much as possible to $\prod_{i=1}^{n} p^*(X_i \mid X_{F_i^*})$ (which is equal to $p^*(X_1, ..., X_n)$), where F_i is the parent node set of node v_i in G and X_{F_i} is the variables of the infection statuses of v_i's parent nodes in G. To achieve this, we adopt relative entropy (*a.k.a.* Kullback-Leibler divergence) to measure the divergence between the above two probability distributions. Formally, the relative entropy can be calculated as follows.

$$\begin{aligned}
&KL(p^*, p)\\
&= \prod_{i=1}^{n} p^*(X_i \mid X_{F_i^*}) \times \log \frac{\prod_{i=1}^{n} p^*(X_i \mid X_{F_i^*})}{\prod_{i=1}^{n} p(X_i \mid X_{F_i})}\\
&= \sum_{x_1,\ldots,x_n} \left(\begin{array}{l} p^*(X_1 = x_1, \ldots, X_n = x_n) \\ \times \log \frac{p^*(X_1 = x_1, \ldots, X_n = x_n)}{\prod_{i=1}^{n} p(X_i = x_i | X_{F_i} = \pi_i)} \end{array} \right)\\
&= \sum_{x_1,\ldots,x_n} \left(\begin{array}{l} p^*(X_1 = x_1, \ldots, X_n = x_n) \\ \times \log p^*(X_1 = x_1, \ldots, X_n = x_n) \end{array} \right)\\
&\quad - \sum_{x_1,\ldots,x_n} \left(\begin{array}{l} p^*(X_1 = x_1, \ldots, X_n = x_n) \\ \times \log \left(\prod_{i=1}^{n} p(X_i = x_i | X_{F_i} = \pi_i) \right) \end{array} \right),
\end{aligned} \tag{2}$$

where x_i is the infection status of node v_i and π_i captures the infection statuses of v_i's parent nodes. The infection status x_i of v_i has two possible values (0 or 1). Moreover, as node v_i has $|F_i|$ parent nodes, there are $2^{|F_i|}$ possible combinations of the infection statuses of v_i's parent nodes.

We estimate the true joint probability distribution $p^*(X_1, ..., X_n)$ by calculating a joint probability distribution $p(X_1, ..., X_n)$ based on the observed node infection status results S. Then, with the following definition of the joint entropy of variables $X_1, \ldots, X_n$,

$$\begin{aligned}
&H(X_1, \ldots, X_n)\\
&= - \sum_{x_1,\ldots,x_n} \left(\begin{array}{l} p(X_1 = x_1, \ldots, X_n = x_n) \\ \times \log p(X_1 = x_1, \ldots, X_n = x_n) \end{array} \right),
\end{aligned} \tag{3}$$

the relative entropy $KL(p^*, p)$ can be estimated as follows.

$$\begin{aligned}
&KL(p^*, p)\\
&= - H(X_1, \ldots, X_n)\\
&\quad - \sum_{x_1,\ldots,x_n} \left(\begin{array}{l} p(X_1 = x_1, \ldots, X_n = x_n) \\ \times \log \left(\prod_{i=1}^{n} p(X_i = x_i | X_{F_i} = \pi_i) \right) \end{array} \right)
\end{aligned} \tag{4}$$

Let $X_{L_i} = \{X_1, \ldots, X_n\} \backslash \{X_i, X_{F_i}\}$, and denote one possible value of X_{L_i} by $\rho_i = \{x_1, \ldots, x_n\} \backslash \{x_i, \pi_i\}$. Then, the second item on the right-hand side of Eq. (4) can be transformed as follows.

$$\begin{aligned}
&\sum_{x_1,\ldots,x_n} \left(\begin{array}{l} p(X_1 = x_1, \ldots, X_n = x_n) \\ \times \log \left(\prod_{i=1}^{n} p(X_i = x_i | X_{F_i} = \pi_i) \right) \end{array} \right)\\
&= \sum_{x_1,\ldots,x_n} \left(\begin{array}{l} p(X_1 = x_1, \ldots, X_n = x_n) \\ \times \left(\sum_{i=1}^{n} \log p(X_i = x_i | X_{F_i} = \pi_i) \right) \end{array} \right)\\
&= \sum_{i=1}^{n} \sum_{x_1,\ldots,x_n} \left(\begin{array}{l} p(X_1 = x_1, \ldots, X_n = x_n) \\ \times \log p(X_i = x_i | X_{F_i} = \pi_i) \end{array} \right)\\
&= \sum_{i=1}^{n} \sum_{x_1,\ldots,x_n} \left(\begin{array}{l} p(X_i = x_i, X_{F_i} = \pi_i) \\ \times p(X_{L_i} = \rho_i | X_i = x_i, X_{F_i} = \pi_i) \\ \times \log p(X_i = x_i | X_{F_i} = \pi_i) \end{array} \right)\\
&= \sum_{i=1}^{n} \sum_{x_i,\pi_i} \left(\begin{array}{l} \left(\begin{array}{l} p(X_i = x_i, X_{F_i} = \pi_i) \\ \times \log p(X_i = x_i | X_{F_i} = \pi_i) \end{array} \right) \\ \times \sum_{\rho_i} p(X_{L_i} = \rho_i | X_i = x_i, X_{F_i} = \pi_i) \end{array} \right)\\
&= \sum_{i=1}^{n} \sum_{x_i,\pi_i} \left(\begin{array}{l} p(X_i = x_i, X_{F_i} = \pi_i) \\ \times \log p(X_i = x_i | X_{F_i} = \pi_i) \end{array} \right)
\end{aligned} \tag{5}$$

Next, the entropy of X_i conditioned on X_{F_i} is as follows.

$$\begin{aligned}
&H(X_i \mid X_{F_i})\\
&= - \sum_{x_i,\pi_i} \left(\begin{array}{l} p(X_i = x_i, X_{F_i} = \pi_i) \\ \times \log p(X_i = x_i | X_{F_i} = \pi_i) \end{array} \right)\\
&= - \sum_{x_i,\pi_i} \left(\begin{array}{l} \sum_{\ell=1}^{\beta} \frac{1}{\beta} I(x_i^\ell = x_i, \pi_i^\ell = \pi_i) \\ \times \log \frac{\sum_{\ell=1}^{\beta} I(x_i^\ell = x_i, \pi_i^\ell = \pi_i)}{\sum_{\ell=1}^{\beta} I(\pi_i^\ell = \pi_i)} \end{array} \right),
\end{aligned} \tag{6}$$

where $I(\cdot)$ is the indicator function. With this definition, the formulation of relative entropy $KL(p^*, p)$ can be finally simplified as follows.

$$KL(p^*, p) = -H(X_1, \ldots, X_n) + \sum_{i=1}^{n} H(X_i \mid X_{F_i}) \tag{7}$$

Note that in Eq. (7), the computation of joint entropy $H(X_1, \ldots, X_n)$ is independent of the diffusion network structure, but depends only on the observed node infection status results S. Therefore, for a given S, the value of $H(X_1, \ldots, X_n)$ is fixed, and the value of the relative entropy $KL(p^*, p)$ is determined by the parent node set F_i we find for each node v_i in the network. Specifically, a parent node set F_i with smaller $H(X_i \mid X_{F_i})$ value results in a lower relative entropy value.

Put differently, a small $H(X_i \mid X_{F_i})$ value indicates a high probability that the infection of $v_i \in V$ is affected by the nodes in F_i ($F_i \subset V$, $v_i \notin F_i$).

4.2 Consideration of Network Complexity

The comparison of values $H(X_i \mid X_{F_i})$ and $H(X_i \mid X_{F_i'})$ for two different parent node sets F_i and F_i' can help us estimate which parent node set has a higher probability of affecting node v_i's infection. If F_i is a subset of F_i', i.e., $F_i \subseteq F_i'$, the relationship $H(X_i \mid X_{F_i'}) \leqslant H(X_i \mid X_{F_i})$ holds, which can be explained by the following theorem and proof.

Theorem 1. *Assume a diffusion network G with node set V and node infection status results S. Further assume that a node $v \in V$ has infection status variable X and parent node set F with*

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2019.2930060, IEEE Transactions on Knowledge and Data Engineering

infection variables X_F. Then, the value of $H(X|X_F)$ decreases with the growth of F.

Proof. For any node set $W \subseteq V\backslash\{v, F\}$, the following derivation hold.

$$\begin{aligned}
&H\left(X|X_{\{F,W\}}\right) - H\left(X|X_F\right) \\
&= -\sum_{x,f,w} p\left(x,f,w\right)\log p\left(x|f,w\right) + \sum_{x,f} p\left(x,f\right)\log p\left(x|f\right) \\
&= -\sum_{x,f,w} p\left(x,f,w\right)\log p\left(x|f,w\right) \\
&\quad + \sum_{x,f}\left(\sum_{w} p\left(x,f,w\right)\right)\log p\left(x|f\right) \\
&= -\sum_{x,f,w} p\left(x,f,w\right)\log p\left(x|f,w\right) \\
&\quad + \sum_{x,f,w} p\left(x,f,w\right)\log p\left(x|f\right) \\
&= \sum_{x,f,w} p\left(x,f,w\right)\log\frac{p\left(x|f\right)}{p\left(x|f,w\right)},
\end{aligned} \tag{8}$$

where x, f, and w refer to the values of the infection status variables X, X_F, and X_W, respectively.

Since the relationship $\log(y) \leqslant (y-1)$ always holds for any nonnegative real number y, we have

$$\begin{aligned}
&H\left(X|X_{\{F,W\}}\right) - H\left(X|X_F\right) \\
&\leqslant \sum_{x,f,w} p\left(x,f,w\right)\cdot\left(\frac{p\left(x|f\right)}{p\left(x|f,w\right)} - 1\right)
\end{aligned} \tag{9}$$

Moreover, the following formula derivation holds.

$$\begin{aligned}
&\sum_{x,f,w} p\left(x,f,w\right)\cdot\left(\frac{p\left(x|f\right)}{p\left(x|f,w\right)} - 1\right) \\
&= \sum_{x,f,w} p\left(x,f,w\right)\frac{p\left(x|f\right)}{p\left(x|f,w\right)} - \sum_{x,f,w} p\left(x,f,w\right) \\
&= \sum_{x,f,w} p\left(x|f,w\right)p\left(f,w\right)\frac{p\left(x|f\right)}{p\left(x|f,w\right)} - 1 \\
&= \sum_{x,f,w} p\left(f,w\right)p\left(x|f\right) - 1 \\
&= \sum_{x,f}\left(\sum_{w} p\left(f,w\right)\right)p\left(x|f\right) - 1 \\
&= \sum_{x,f} p\left(f\right)p\left(x|f\right) - 1 \\
&= 1 - 1 \\
&= 0
\end{aligned} \tag{10}$$

Therefore, the relationship

$$H\left(X|X_{\{F,W\}}\right) - H\left(X|X_F\right) \leqslant 0 \tag{11}$$

always hold, and the theorem is correct. □

According to Theorem 1, the minimum of $H(X_i|X_{F_i})$ is obtained after adding all other nodes to the parent node set F_i. The reason is that each other node may possibly affect the infection of node v_i. And we have no prior knowledge on v_i's parent nodes. However, if we take into account only relative entropy to infer influence relationships and simply pursue a lower value of $H(X_i|X_{F_i})$, the result will be a very complex graph G that contains many low-probability influence relationships, many of which may not exist in reality. Further, the resulting graph G will also be computationally and conceptually difficult to use in practice.

On the other hand, if we include more nodes into the parent node set F_i of v_i for each $v_i \in V$, this yields larger statistical errors when computing $H(X_i|X_{F_i})$. The reason is that given an F_i then for every possible combination π_{ij} of the infection statuses of the nodes in F_i (where π_{ij} refers to the j-th possible combination, $1 \leqslant j \leqslant 2^{|F_i|}$), we need to find its instantiations from all the β historical diffusion processes to estimate the corresponding probabilities $p(X_i = x_i, X_{F_i} = \pi_{ij})$ and $p(X_i = x_i|X_{F_i} = \pi_{ij})$ (where $x_i \in \{0, 1\}$) in order to compute $H(X_i|X_{F_i})$. Therefore, the number of probability estimations increases exponentially with the cardinality of F_i. In each of these probability estimations, an error may be introduced when the number of corresponding instantiations is insufficient. For a fixed number β of historical diffusion processes, the more probability estimations to make, the fewer the average available instantiations for each estimation, resulting in larger statistical errors. In brief, the introduced statistical errors are affected by two factors: (1) the number $2^{|F_i|}$ of possible infection status combinations of the nodes in each F_i and (2) the number β of historical diffusion processes to be used.

To balance the relative entropy and the network complexity and to reduce the introduction of statistical errors, we combine the above two factors and add a penalty term $\lambda 2^{|F_i|}$ to $H(X_i|X_{F_i})$, where $\lambda \geqslant 0$ is a function of β. Then, we have a selection criterion $g(v_i, F_i)$ for the parent node set F_i of node $v_i \in V$, which can be formulated as follows.

$$g(v_i, F_i) = 2H(X_i|X_{F_i}) + \lambda 2^{|F_i|}. \tag{12}$$

A smaller value of $g(v_i, F_i)$ indicates that the current F_i is a better parent node set selection for node v_i. According to the above selection criterion, inclusion of more parent nodes for each node v_i will decrease the value of $2H(X_i|X_{F_i})$, but will also result in an exponential increase in the value of the penalty term $\lambda 2^{|F_i|}$, which will help us avoid adding too much nodes into set F_i. Furthermore, to make the F_i selected by $g(v_i, F_i)$ as consistent as possible with the true parent node set F_i^* of node v_i, the λ used in $g(v_i, F_i)$ should satisfy the conditions $\lim_{\beta\to\infty}\frac{\lambda}{\beta} = 0$ and $\lim_{\beta\to\infty}\frac{\lambda}{\log\log\beta} = +\infty$. This is explained by the following corollary, which follows directly from Theorem 4 in reference [25].

Corollary 1. *Let $\hat{F}_i$ be the parent node set selected by the criterion $g(v_i, F_i) = 2H(X_i|X_{F_i}) + \lambda 2^{|F_i|}$ for a given node $v_i \in V$ based on infection status results S of β historical diffusion processes, i.e., $\hat{F}_i$ minimizes the value of $g(v_i, F_i)$ for a given v_i. If λ satisfies conditions*

$$\lim_{\beta\to\infty}\frac{\lambda}{\beta} = 0, \ \textit{and} \ \lim_{\beta\to\infty}\frac{\lambda}{\log\log\beta} = +\infty, \tag{13}$$

then $\hat{F}_i$ is a strongly consistent estimator of the true parent node set F_i^* of node v_i, i.e.,

$$\lim_{\beta\to\infty}\hat{F}_i = F_i^* \tag{14}$$

Based on Corollary 1, we set the value of λ to $\log\beta$.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2019.2930060, IEEE Transactions on Knowledge and Data Engineering

To obtain an optimal F_i that minimizes the value of criterion $g(v_i, F_i)$, one should intuitively find a few parent nodes that are most likely to affect the infection of node v_i, and one should prevent the set of parent nodes from growing too large. In fact, upper bound for the number of parent nodes can be derived from the selection criterion.

Theorem 2. *To minimize the value of $g(v_i, F_i)$, the size of set F_i cannot exceed* $\log \frac{2\beta}{\log \beta}$.

Proof. Assuming that we have already found a parent node set F_i for node v_i, let node set $R_i = V \setminus \{F_i, v_i\}$ be the remaining nodes after removing F_i and v_i from V. Then if we add a node $v_j \in R_i$ and any node set $W \subseteq R_i \setminus \{v_j\}$ to F_i, the following relationships hold.

$$\begin{aligned} & g\left(v_i, \{F_i, v_j, W\}\right) \\ & = 2H(X_i|X_{\{F_i,v_j,W\}}) + 2^{|\{F_i,v_j,W\}|} \log \beta \\ & \geqslant 2H(X_i|X_{\{F_i,R_i\}}) + 2^{|\{F_i,v_j,W\}|} \log \beta \\ & \geqslant 2H(X_i|X_{\{F_i,R_i\}}) + 2^{|\{F_i,v_j\}|} \log \beta \end{aligned} \tag{15}$$

If the value of $2H(X_i|X_{\{F_i,R_i\}}) + 2^{|\{F_i,v_j\}|} \log \beta$ exceeds the current $g(F_i)$, then we have that $g(v_i, \{F_i, v_j, W\}) > g(v_i, F_i)$, which indicates that adding node v_j or any node set $\{v_j, W\}$ to the current F_i will increase the value of $g(v_i, F_i)$. Therefore, if a node $v_j \in R_i$ can be added to the parent node set F_i of node v_i, the following prerequisite must be met.

$$\begin{aligned} & 2H(X_i|X_{\{F_i,R_i\}}) + 2^{|\{F_i,v_j\}|} \log \beta \\ & = 2H(X_i|X_{\{F_i,R_i\}}) + 2^{|F_i|+1} \log \beta \\ & \leqslant g(v_i, F_i). \end{aligned} \tag{16}$$

Moreover, the following relationship holds.

$$\begin{aligned} g(v_i, F_i) &= 2H(X_i \mid F_i) + 2^{|F_i|} \log \beta \\ &\leqslant 2H(X_i \mid \emptyset) + 2^{|F_i|} \log \beta \end{aligned} \tag{17}$$

Combining this with the prerequisite, we get:

$$\begin{aligned} & 2H(X_i \mid \emptyset) + 2^{|F_i|} \log \beta \\ & \geqslant 2H(X_i|X_{\{F_i,R_i\}}) + 2^{|F_i|+1} \log \beta, \end{aligned} \tag{18}$$

which can be simplified as

$$2^{|F_i|} \log \beta \leqslant 2H(X_i \mid \emptyset) - 2H(X_i|X_{\{F_i,R_i\}}) \tag{19}$$

According to the definition of $H(X_i|\emptyset)$ and $H(X_i|X_{\{F_i,R_i\}})$, relationships $H(X_i|\emptyset) \leqslant \beta$ and $H(X_i|X_{\{F_i,R_i\}}) \geqslant 0$ hold. Combining them with the inequality above, we obtain

$$2^{|F_i|} \log \beta \leqslant 2\beta \tag{20}$$

This indicates that if any new parent node can be added to the current F_i to decrease the value of $g(v_i, F_i)$, the number $|F_i|$ of nodes in current F_i should satisfy

$$|F_i| \leqslant \log \frac{2\beta}{\log \beta} \tag{21}$$

In other words, the size of of set F_i cannot exceed $\log \frac{2\beta}{\log \beta}$, and the theorem is correct. □

Given the selection criterion $g(v_i, F_i)$ and Theorem 2, we can apply a greedy search procedure to find the most probable parent nodes for v_i. The procedure starts from an empty parent node set F_i, and expands F_i by iteratively adding a node combination (i.e., a subset of $V \setminus \{v_i\}$) that decreases the value of the current $g(v_i, F_i)$ the most. The procedure stops when the number of nodes in F_i reaches the upper bound (i.e., $|F_i| \geqslant \log \frac{2\beta}{\log \beta}$) or no candidate parent node for v_i exists. This way, we can efficiently achieve a locally optimal F_i. A similar greedy search procedure is used commonly in many other applications, such as influence maximization [29] and classification [32], due to its efficiency and good result quality.

4.3 Pruning of Candidate Parent Nodes

To find for each node $v_i \in V$ a candidate parent node or node set that can be added to its current parent node set F_i during the greedy search procedure, a straightforward method is to traverse all node combinations containing at most $\log \frac{2\beta}{\log \beta}$ nodes from the candidate parent node set $V \setminus \{v_i\}$. This straightforward method is inefficient since there are $\sum_{i=1}^{\log(2\beta/\log\beta)} \binom{i}{n-1}$ combinations, where n is the number of nodes in the network. Instead, we prune the candidate parent nodes to reduce the number of possible node combinations and to avoid redundant computations during the greedy search procedure.

In order to minimize the value of $g(v_i, F_i)$, a node v_k with a small value of $H(X_i|X_k)$ is more likely to be an appropriate candidate parent node of node v_i, compared with a node v_j with a large value of $H(X_i|X_j)$. Observe that for each node $v_j \in V$, the upper bound of value $H(X_i|X_j)$ is $H(X_i)$, i.e., $H(X_i|X_j) \leqslant H(X_i)$. Then, if the value of $H(X_i|X_j)$ is large and close to its upper bound $H(X_i)$, this indicates that the uncertainty of variable X_i is almost unchanged regardless of whether we take into account variable X_j. Put differently, the infection statuses of the two nodes v_i and v_j are independent or have extremely low correlation. In information theory, the mutual information, abbreviated as MI, of variables X_i and X_j is calculated as follows.

$$\begin{aligned} MI(X_i, X_j) &= H(X_i) - H(X_i|X_j) \\ &= p(X_i, X_j) \log \frac{p(X_i, X_j)}{p(X_i)p(X_j)} \end{aligned} \tag{22}$$

The mutual information is the difference between entropy $H(X_i)$ and conditional entropy $H(X_i|X_j)$. A value of $MI(X_i, X_j)$ close to 0 indicates that there is a very low probability that nodes v_i and v_j have an influence relationship with each other. Furthermore, in a real-world diffusion network, each node v_i often has a finite number of parent nodes, and most other nodes in the network have no influence relationship with v_i. Therefore, most of the MI values between the infection statuses of v_i and its candidate parent nodes are very close to 0. These small MI values form a compact cluster with a very small mean (also close to 0).

Inspired by this line of reasoning, we introduce a heuristic pruning method based on the MI values with the goal of screening out insignificant candidate parent nodes for each node. By performing a modified K-means algorithm with $K = 2$ and one of the two means fixed at 0 through all iterations of K-means, we can efficiently partition all MI values into two groups, where one group has a mean very close to 0. Let τ be the largest value in the cluster with mean close to 0. Then, for each $MI(X_i, X_j) \leqslant \tau$, we regard

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2019.2930060, IEEE Transactions on Knowledge and Data Engineering

Algorithm 1: The SIDN 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 $E \leftarrow \emptyset$; // set of inferred directed edges
2 **for** each $v_i \in V$ **do**
3 **for** each $v_j \in V (j \neq i)$ **do**
4 Calculate MI value $MI(X_i, X_j)$ using Eq. (22);
5 Partition all MI values into two groups by K-means (with $K = 2$ and one mean fixed at 0) and set τ to the largest value in the group with mean close to 0;
6 **for** each $v_i \in V$ **do**
7 $P_i \leftarrow \emptyset$; // v_i's candidate parent node set
8 $C_i \leftarrow \emptyset$;//v_i's possible parent node combination set
9 $F_i \leftarrow \emptyset$; // v_i's inferred parent node set
10 **for** each $v_j \in V (j \neq i)$ **do**
11 **if** $MI(X_i, X_j) > \tau$ **then**
12 $P_i \leftarrow P_i \bigcup \{v_j\}$; // insert v_j into P_i
13 **for** each $W \subseteq P_i$, $|W| \leqslant \log \frac{2\beta}{\log \beta}$ **do**
14 Calculate $g(v_i, W)$ using Eq. (12);
15 $C_i \leftarrow \{C_i, W\}$; // add a new element W to C_i
16 **while** $C_i \neq \emptyset$ **do**
17 **if** $|F_i| < \log \frac{2\beta}{\log \beta}$ **then**
18 $W^* \leftarrow \arg\min_{W \in C_i} g(v_i, W)$;
19 $F_i \leftarrow F_i \bigcup W^*$;
20 $C_i \leftarrow C_i \backslash W^*$;
21 **else**
22 **break**;
23 $E \leftarrow \{(v_j, v_i) \mid v_j \in F_i\} \bigcup E$; // (v_j, v_i) is directed

the corresponding node v_j as an insignificant candidate parent node for v_i and exclude v_j from the candidate parent node set of v_i. This pruning method allows us to screen out insignificant candidate parent nodes, thus allowing the SIDN algorithm to focus on parent node combinations that are more likely to exist in the real network.

4.4 Algorithm

To find the most probable influence relationships, we introduce a probabilistic generative model based on relative entropy. By taking into account the network complexity, we design a selection criterion to balance the relative entropy and the complexity of inferred influence relationships, and we simultaneously avoid introducing large statistical errors. To eliminate redundant computations during the influence relationship inferencing, we present a heuristic pruning method to screen out insignificant candidate parent nodes. Based on these computational elements, we propose the SIDN algorithm for the problem of learning diffusion network structure with only the infection status information.

The SIDN algorithm, outlined in Algorithm 1, takes as inputs node set V of the objective diffusion network G and a set S of infection status results observed on V across β diffusion processes. It first initializes the inferred directed edge set E of G as an empty set (line 1), following by calculating the MI value for each node pair (lines 2–4), and performing the modified K-means algorithm on all the MI values (with $K = 2$ and one mean fixed at 0 through all K-means iterations) to find a MI threshold τ (line 5), which is used to screen out insignificant candidate parent nodes. Then, the algorithm infers the incoming edges to each node $v_i \in V$ by the following five steps: (1) Firstly, three empty sets P_i, C_i and F_i are initialized to record v_i's candidate parent nodes, possible parent node combinations and inferred parent nodes, respectively (lines 7–9). (2) Secondly, for each node $v_j \in V (j \neq i)$ (line 10), if the corresponding value of $MI(X_i, X_j)$ is larger than MI threshold τ (line 11), then the node v_j will be inserted into the candidate parent node set P_i of v_i (line 12), otherwise it will be regarded as an insignificant candidate parent node of v_i. (3) Thirdly, for each possible parent node combination $W \subseteq P_i$ with size less than $\log \frac{2\beta}{\log \beta}$ (line 13), which is the upper bound for the size of a parent node set, the corresponding score $g(v_i, W)$ is calculated and recorded (line 14), and the node combination W is added into possible parent node combination set C_i as a new element (line 15). (4) Fourthly, the inferred parent node set F_i will be continuously expanded with the parent node combination $W^* \in C_i$ that has the currently smallest value of $g(v_i, W)$ ($W \in C_i$) until the size of F_i reaches the upper bound $\log \frac{2\beta}{\log \beta}$ or no more candidate parent node combinations exist (lines 16–22). (5) Finally, a directed edge from each node in F_i to v_i is added to the inferred edge set E of the objective diffusion network G (line 23).

4.5 Complexity Analysis

The most computationally expensive process in SIDN consists of the following two parts. (1) To disqualify insignificant candidate parent nodes, calculating mutual information values requires $O(\beta n^2)$ time, and performing K-means clustering on the mutual information values takes $O(tn^2)$ time, where n is the number of nodes in the network, β is the number of diffusion processes, and t is the number of K-means iterations ($t \ll n$). (2) To find candidate parent node sets, calculating all relative entropy values requires $O(\eta^2 \kappa^\eta n \beta)$ time, since there are at most $\sum_{i=1}^{\eta} \binom{i}{\kappa} < \eta \kappa^\eta$ candidate parent node combinations for each node and calculating the relative entropy for each candidate parent node combination takes at most $O(\beta\eta)$ time, where η denotes the upper bound of parent node set size (i.e., $\eta = \log \frac{2\beta}{\log \beta}$), κ denotes the maximum number of candidate parent nodes for each node. Since most candidate parent nodes are insignificant (discussed in Section 4.3), after screening out these nodes with the proposed MI-based pruning method, κ is usually much smaller than n, i.e., $\kappa \ll n$.

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

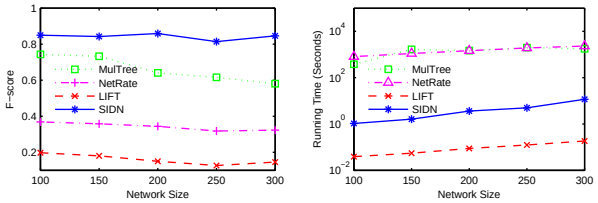
5 Experimental Evaluation

We first introduce the experimental setup, and then report on experiments designed to gain insight into the effectiveness and efficiency of the SIDN algorithm on both synthetic and real-world networks. To this end, we investigate the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2019.2930060, IEEE Transactions on Knowledge and Data Engineering

TABLE 2
Properties of LFR Benchmark Graphs used for Experiments.

Graphs	n	$\mathcal{K}$	$\mathcal{T}$
LFR1-5	100,150,200,250,300	4	2
LFR6-10	200	2,3,4,5,6	2
LFR11-15	200	46	1,1.5,2,2.5,3



(a) *F-score* (b) *Running Time*

Fig. 1. Effect of Diffusion Network Size

effects of diffusion network size, the average node degree, the degree dispersion of the diffusion network, the initial infection ratio, the transmission rates (i.e., the propagation probabilities between nodes), the number of diffusion processes, the MI-based pruning method, and the network complexity consideration on the accuracy and running time of SIDN. All algorithms are implemented in Java, running on a desktop PC with an Intel Core i3-6100 CPU at 3.70GHz and 8GB RAM.

5.1 Experimental Setup

Networks. We adopt the LFR benchmark graphs [17] as the synthetic networks. By using different graph generation parameters, such as the number n of nodes, the average degree $\mathcal{K}$ of each node, and the degree distribution parameter $\mathcal{T}$ (larger $\mathcal{T}$ implies less dispersion of degrees), we generate three series of graphs with properties summarized in Table 2. In addition, we adopt two real-world networks, i.e., NetSci [24], which is a coauthorship network containing 379 scientists and 1602 coauthorships, and DUNF [31], which is a microblogging network with 750 users and 2974 following relationships, for the experimental evaluation.

Infection Data. The infection status results S can be obtained by simulating β diffusion processes on each network with randomly selected initially infected nodes in each simulation (the initial infection ratio is α). Corresponding cascades are also recorded for tested cascade-based algorithms in the experiments. In each diffusion process, each infected node tries to infect its uninfected child nodes with a given transmission rate, which is subjected to a Gaussian distribution with mean μ and variance 0.05, to ensure that more than 95% of all transmission rate values are within the range from $\mu - 0.1$ to $\mu + 0.1$.

Performance Criteria. To evaluate the accuracy of the SIDN algorithm on the inference of diffusion network structure, we report the F-score (i.e., the harmonic mean of precision and recall) of its inferred directed edges, which can be calculated as $F\text{-}score = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall}$, where $Precision = \frac{N_{TP}}{N_{TP}+N_{FP}}$ and $Recall = \frac{N_{TP}}{N_{TP}+N_{FN}}$. Here, N_{TP} denotes the number of true positives, i.e., the edges in the real network that are inferred correctly by the algorithm; N_{FP} denotes the number of false positives, i.e., edges that do not exist in the real network, but that are inferred falsely by the algorithm; and N_{FN} denotes the number of false negatives, i.e., edges that exist in the real network, but that are not inferred by the algorithm.

Benchmark Algorithms. Among the existing infection timestamp-based algorithms, embedding-based methods do not infer an explicit diffusion network structure. Therefore, we compare our algorithm with the state-of-the-art convex programming-based approach NetRate [7] and the high performance submodularity-based algorithm MulTree [11]. In addition, as the PATH algorithm [12] requires all path connected node triples, which are difficult to obtain in practice, we choose the infection timestamp-free approach LIFT [2] for comparison. Since NetRate infers the transmission rate between each two nodes in the network, we give NetRate a preferential treatment in accuracy comparisons. Specifically, when calculating the F-scores of edges whose transmission rates exceed a threshold, we use different thresholds to find the highest F-score and report it as the accuracy of NetRate. Moreover, since MulTree and LIFT need users to specify the number of edges to be inferred, we provide the real number m of edges in the network to these two algorithms.

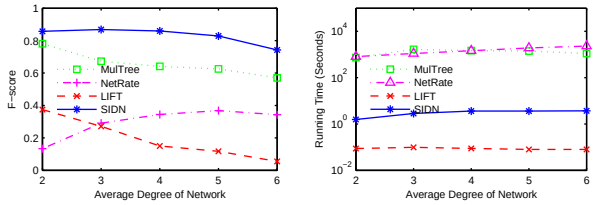
5.2 Effect of Diffusion Network Size

To study the effect of diffusion network size on algorithm performance, we adopt five synthetic networks, i.e., LFR1–5, where the size varies from 100 to 300. We simulate 150 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., $\alpha = 0.15$), and the mean μ of transmission rates is set to 0.3.

Fig. 1 reports the F-score and running time of each algorithm, from which we can observe that (1) a larger diffusion network size tends to degrade the accuracy of NetRate, LIFT, and MulTree, while the accuracy of SIDN is reasonably insensitive to the diffusion network size and outperforms the other algorithms. (2) The running time of each algorithm increases with the diffusion network size. LIFT executes the fastest (but with low accuracy), and SIDN is an order of magnitude faster (and has higher accuracy) than MulTree and NetRate.

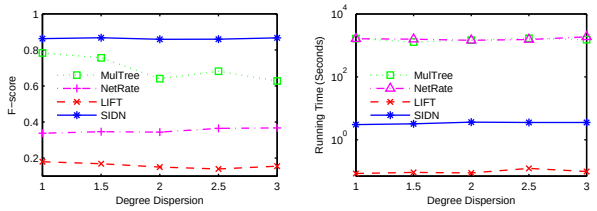
In addition, based on extensive testing on larger networks, we have found that the running time of NetRate and MulTree rapidly increase with the growth of network size. Even with a medium network size, their running time exceeds acceptable levels, and meanwhile, the execution of SIDN starts to witness out-of-memory errors (given these facts, we have selected and tested networks with sizes varying from 100 to 750). The reason behind is that NetRate and MulTree consider all propagation paths supported by each cascade without any pruning strategy, while SIDN records the MI value for each node pair for its MI-based pruning method. Thus, if the objective network contains more nodes, there will be significantly more possible propagation paths and relatively more node pairs, incurring higher computational costs for NetRate and MulTree and a larger space requirement for SIDN.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2019.2930060, IEEE Transactions on Knowledge and Data Engineering



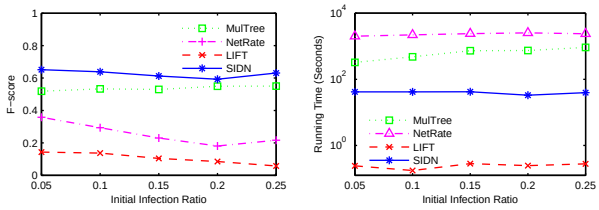
(a) *F-score* (b) *Running Time*

Fig. 2. Effect of Average Node Degree



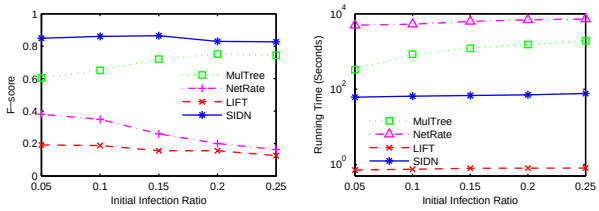
(a) *F-score* (b) *Running Time*

Fig. 3. Effect of Node Degree Dispersion of Diffusion Network



(a) *F-score* (b) *Running Time*

Fig. 4. Effect of Initial Infection Ratio on NetSci



(a) *F-score* (b) *Running Time*

Fig. 5. Effect of Initial Infection Ratio on DUNF

5.3 Effect of Average Node Degree

The edge density of diffusion network can affect the number of influence relationships. The average node degree, i.e., the total number of edges divided by the total number of nodes, is usually used to represent the edge density of a network.

To study the effect of a network's average degree on algorithm performance, we test the algorithms on five synthetic networks, i.e., LFR6–10, where the average degree varies from 2 to 6. We simulate 150 diffusion processes on each network (i.e., $\beta = 150$). In each simulation, $0.15n$ nodes are randomly selected as the initially infected nodes (i.e., $\alpha = 0.15$), and the mean μ of transmission rates is set to 0.3.

Fig. 2 reports the F-score and running time of each algorithm, from which we can observe that (1) as the average degrees of diffusion networks increase, accuracy of MulTree, SIDN, and LIFT decrease. The accuracy of NetRate increases when the average degree increases from 2 to 5 and then decreases when the average degree reaches 6. Compared with the other tested algorithms, the SIDN algorithm has the best accuracy. (2) The running times of MulTree, NetRate, and SIDN increase with the growth of average degree, and SIDN shows a significant running time advantage over MulTree and NetRate.

5.4 Effect of Node Degree Dispersion

If a diffusion network has a large degree dispersion, i.e., different nodes have different numbers of edges, then there will be variations in the influence diffusion capabilities of different parts of the network, which can affect the diffusion processes and the final infection statuses of nodes.

To study the effect of the node degree dispersion on algorithm performance, we test the algorithms on five synthetic networks, i.e., LFR11–15, where the degree distribution parameters vary from 1 to 3 (the corresponding standard deviation of the degree varies from about 0.8 to about 0.4). We simulate 150 diffusion processes on each network (i.e., $\beta = 150$). In each of these simulations, $0.15n$ nodes are randomly selected as the initially infected nodes (i.e., $\alpha = 0.15$), and the mean transmission rate μ is set to 0.3.

Fig. 3 reports the F-score and running time of each algorithm, from which we can observe that (1) an increase in the degree distribution parameter tends to reduce the accuracy of MulTree. The accuracy of NetRate, LIFT, and SIDN is reasonably insensitive to degree dispersion, and SIDN performs better than other algorithms. (2) Degree dispersion has little effect on the running times of the algorithms, and SIDN has better running time performance than NetRate and MulTree, and LIFT is fastest.

5.5 Effect of Initial Infection Ratio

The ratio of initially infected nodes may affect the number of final infected nodes in a diffusion process.

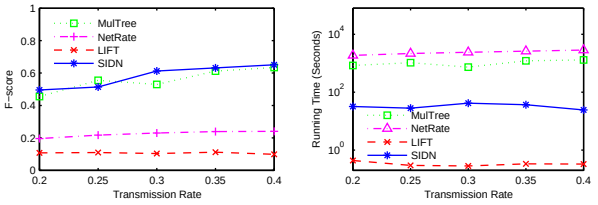
To study the effect of the initial infection ratio on performance, we test the algorithms on real-world networks NetSci and DUNF with different initial infection ratios α (varied from 0.05 to 0.25). For each initial infection ratio, we simulate 150 diffusion processes on each network (i.e., $\beta = 150$) with the mean transmission rate μ fixed at 0.3.

Figs. 4–5 report the F-score and running time of each algorithm on NetSci and DUNF, repectively. From the figures, we can observe that an increase of initial infection ratio tends to improve the accuracy of MulTree, but degrades the accuracy of LIFT and NetRate. SIDN is reasonably insensitive to variations in the initial infection ratio and has the best accuracy. Further, an increase in the initial infection ratio has little effect on the running time of SIDN and LIFT, but results in longer running time for MulTree and NetRate. Similar results can also be observed on the synthetic networks LRF1–15.

5.6 Effect of Transmission Rate

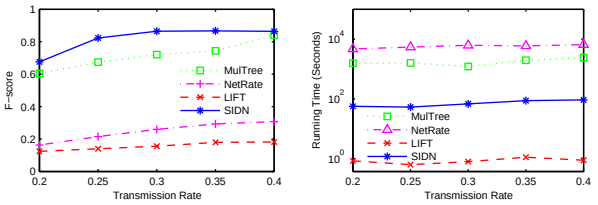
The transmission rates between nodes may affect the correlations between the infections of parent nodes and cor-

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2019.2930060, IEEE Transactions on Knowledge and Data Engineering



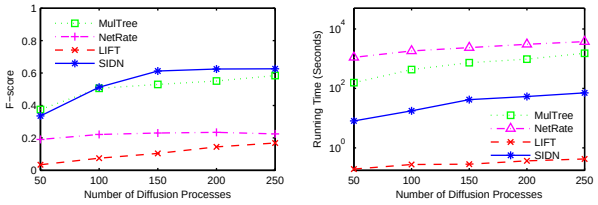
(a) *F-score* (b) *Running Time*

Fig. 6. Effect of Transmission Rate on NetSci



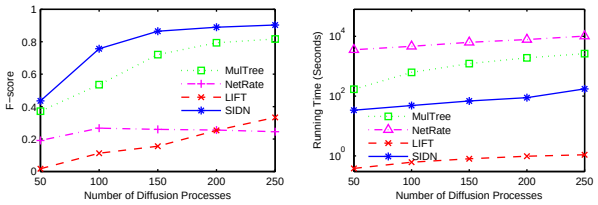
(a) *F-score* (b) *Running Time*

Fig. 7. Effect of Transmission Rate on DUNF



(a) *F-score* (b) *Running Time*

Fig. 8. Effect of Number of Diffusion Processes on NetSci



(a) *F-score* (b) *Running Time*

Fig. 9. Effect of Number of Diffusion Processes on DUNF

responding child nodes. Therefore, the transmission rates may affect the accuracy of diffusion network inference. Generally, higher transmission rates are expected to enhance the correlations between the observed infection statuses of parent nodes and corresponding child nodes, and they will likely help the inference algorithms identify influence relationships between nodes more effectively, resulting in an accuracy improvement for the algorithms.

To study the effect of the transmission rate on algorithm performance, we test the algorithms on real-world networks NetSci and DUNF with different transmission rate settings, where we vary the mean transmission rate μ from 0.2 to 0.4. For each transmission rate setting, we simulate 150 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., $\alpha = 0.15$).

Figs. 6–7 report the F-score and running time of each algorithm on NetSci and DUNF, repectively. We can observe that the accuracy and running time of each algorithm increase as the transmission rate increases. Further, SIDN generally achieves the best accuracy, with MulTree slightly better in one setting and being close for some other settings. The running times are similar to what is observed in pervious experiments. It is also noted that similar results can be observed on synthetic networks LRF1–15.

5.7 Effect of The Number of Diffusion Processes

The inference of a diffusion network is based on the observed results of diffusion processes. Hence, the number of diffusion processes may affect the accuracy of diffusion network inference. Generally, more diffusion processes will expose more information about a diffusion network, and this may help diffusion network inference algorithms achieve more accurate inference results.

To study the effect of the amount of diffusion processes on algorithm performance, we test the algorithms on 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 select $0.15n$ nodes as the initially infected nodes ($\alpha = 0.15$), and the mean transmission rate μ is set to 0.3.

Figs. 8–9 show the F-score and running time of each algorithm on NetSci and DUNF, repectively. We can observe that a larger number of diffusion processes often helps the algorithms achieve more accurate results on network structure inference. SIDN achieves the best accuracy when compared with the other algorithms in all but one setting. To analyze the infection statuses created by more diffusion processes, the algorithms generally require longer running time. Compared with MulTree and NetRate, SIDN shows a significant advantage in terms of running time, while LIFT has the lowest running time. Similar results can also be observed on synthetic networks LRF1–15.

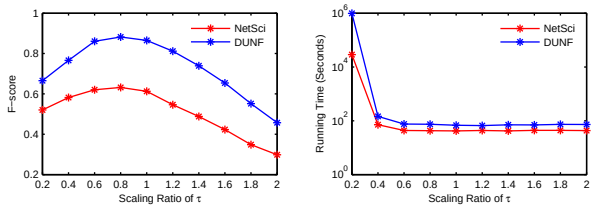
5.8 Effect of MI-based Pruning Method

To screen out insignificant candidate parent nodes and eliminate redundant computations during the influence relationship inferencing, SIDN adopts a MI-based pruning method, which finds a MI threshold τ for the identification of insignificant candidate parent nodes.

To study the effect of the MI-based pruning method on the performance of SIDN, we test SIDN on real-world networks NetSci and DUNF with different MI thresholds. Since the running time of SIDN with a MI threshold equal to 0 (i.e., SIDN without candidate parent node pruning) on the networks is prohibitively long and beyond acceptable, we omit to report the corresponding performance results. We vary the MI threshold from 0.2τ to 2τ, and for each MI threshold, we simulate 150 diffusion processes on each network (i.e., $\beta = 150$) with $0.15n$ initially infected nodes that are randomly selected in each simulation (i.e., $\alpha = 0.15$) and the mean transmission rate μ fixed at 0.3.

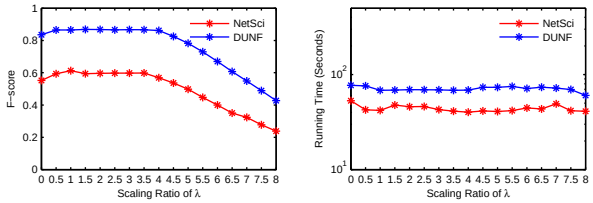
Fig. 10 reports the F-score and running time of SIDN with different MI thresholds. We can observe that the MI

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2019.2930060, IEEE Transactions on Knowledge and Data Engineering



(a) *F-score* (b) *Running Time*

Fig. 10. Effect of MI-based Pruning Method



(a) *F-score* (b) *Running Time*

Fig. 11. Effect of Network Complexity Consideration

threshold τ found by the MI-based pruning method is able to help SIDN achieve a nearly optimal accuracy. When the MI threshold is less than 0.6τ, the smaller MI threshold the lower accuracy of SIDN. When the MI threshold is more than τ, the larger MI threshold the lower accuracy of SIDN. This is because a smaller MI threshold has a weaker effect of pruning, and thus leaves more insignificant candidate parent nodes for parent node selection, causing precision degradation for SIDN; in contrast, if the MI threshold is too large, it may screen out the real candidate parent nodes, resulting in a lower recall for SIDN. Further, compared with using a small MI threshold less than 0.6τ, using the MI threshold τ found by the MI-based pruning method markedly reduces the running time of SIDN. Similar results can also be observed on synthetic networks LRF1–15.

5.9 Effect of Network Complexity Consideration

To avoid overly complex inferred structures and reduce the introduction of statistical errors, SIDN adopts a parent node selection criterion $g(v_i, F_i)$ that takes into account network complexity by using a penalty term $\lambda 2^{|F_i|}$. A larger penalty term coefficient λ generally implies less statistical errors but a more simplified inference result. Based on Corollary 1, we set the penalty term coefficient λ to $\log \beta$.

To study the effect of the network complexity consideration in SIDN, we test SIDN on real-world networks NetSci and DUNF with different penalty term coefficients (varied from 0 to 8λ). For each penalty term coefficient, we simulate 150 diffusion processes on each network (i.e., $\beta = 150$) with $0.15n$ initially infected nodes that are randomly selected in each simulation (i.e., $\alpha = 0.15$) and the mean transmission rate μ fixed at 0.3.

Fig. 11 reports the F-score and running time of SIDN with different penalty term coefficients. We can observe that compared with no consideration of network complexity (penalty term coefficient equal to 0) and an excessive consideration of network complexity (penalty term coefficient larger than 4λ), a reasonable network complexity consideration with penalty term coefficient scaling from 0.5λ to 4λ is able to help SIDN achieve a better accuracy. Moreover, the network complexity consideration does not significantly affect the running time of SIDN. Similar results can also be observed on synthetic networks LRF1–15.

6 Conclusion

In this paper, we have investigated the problem of diffusion network inference without reliance on infection timestamps of nodes. In order to learn the structure, or edges, of a diffusion network based only on observed final node infection statuses following a set of diffusion processes, we have developed a relative entropy-based probabilistic generative model to find potential influence relationships that are most likely to have generated the node infection statuses. Based on the model, we have designed a selection criterion to find the most probable parent nodes for each node in the network by taking into account the complexity of inferred diffusion network structure. Furthermore, we have presented a heuristic pruning method for candidate parent nodes that eliminates redundant computations and improves running time. Extensive experimental results have verified the effectiveness and efficiency of SIDN.

So far we have applied our SIDN algorithm to infer influence relationships in diffusion networks with static structures. To this end, SIDN focuses on the final statuses of nodes at the end of each diffusion process. Nonetheless, if the node set or edge set changes over time, it will miss many intermediate influence relationships and fail to recover a dynamic diffusion network. For the next stage of study, a promising direction is to extend our method to handle with dynamic diffusion networks.

Acknowledgments

This work was supported in part by the National Key R&D Program of China (2018YFB1004003), NSFC Grants (61502347, 61522208 and U1609217), the Technological Innovation Major Projects of Hubei Province (2017AAA125), the Science and Technology Program of Wuhan City (2018010401011288), and Xiaomi-WHU AI Lab. Yunjun Gao is the corresponding author of the work.

References

[1] B. Abrahao, F. Chierichetti, and R. Kleinberg. Trace complexity of network inference. In *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Min.*, 2013, pages 491–499.

[2] K. Amin, H. Heidari, and M. Kearns. Learning from contagion(without timestamps). In *Proc. Int. Conf. Mach. Learn.*, 2014, pages 1845–1853.

[3] S. Bourigault, C. Lagnier, S. Lamprier, L. Denoyer, and P. Gallinari. Learning social network embeddings for predicting information diffusion. In *Proc. ACM Int. Conf Web Search Data Min.*, 2014, pages 393–402.

[4] H. Daneshmand, M. Gomez-Rodriguez, L. Song, and B. Schölkopf. Estimating diffusion network structures: Recovery conditions, sample complexity & soft-thresholding algorithm. In *Proc. Int. Conf. Mach. Learn.*, 2014, pages 793–801.

[5] N. Du, L. Song, A. Smola, and M. Yuan. Learning networks of heterogeneous influence. In *Adv. Neural Inf. Process. Syst.*, 2012, pages 2780–2788.

[6] S. Gao, H. Pang, P. Gallinari, J. Guo, and N. Kato. A novel embedding method for information diffusion prediction in social network big data. *IEEE Trans. Ind. Inform.*, 13(4):2097–2105, 2017.

[7] M. Gomez-Rodriguez, D. Balduzzi, and B. Schölkopf. Uncovering the temporal dynamics of diffusion networks. In *Proc. Int. Conf. Mach. Learn.*, 2011, pages 561–568.

[8] M. Gomez-Rodriguez, J. Leskovec, and A. Krause. Inferring networks of diffusion and influence. In *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Min.*, 2010, pages 1019–1028.

[9] M. Gomez-Rodriguez, J. Leskovec, and B. Schölkopf. Modeling information propagation with survival theory. In *Proc. Int. Conf. Mach. Learn.*, 2013, pages 666–674.

[10] M. Gomez-Rodriguez, J. Leskovec, and B. Schölkopf. Structure and dynamics of information pathways in online media. In *Proc. ACM Int. Conf Web Search Data Min.*, 2013, pages 23–32.

[11] M. Gomez-Rodriguez and B. Schölkopf. Submodular inference of diffusion networks from multiple trees. In *Proc. Int. Conf. Mach. Learn.*, 2012, pages 489–496.

[12] V. Gripon and M. Rabbat. Reconstructing a graph from path traces. In *Proc. IEEE Int. Symp. Inf. Theory*, 2013, pages 2488–2492.

[13] X. He, T. Rekatsinas, J. Foulds, L. Getoor, and Y. Liu. Hawkestopic: A joint model for network inference and topic modeling from text-based cascades. In *Proc. Int. Conf. Mach. Learn.*, 2015, pages 871–880.

[14] X. He, K. Xu, D. Kempe, and Y. Liu. Learning influence functions from incomplete observations. In *Adv. Neural Inf. Process. Syst.*, 2016, pages 2065–2073.

[15] D. Kalimeris, Y. Singer, K. Subbian, and U. Weinsberg. Learning diffusion using hyperparameters. In *Proc. Int. Conf. Mach. Learn.*, 2018, pages 2420–2428.

[16] T. Kurashima, T. Iwata, N. Takaya, and H. Sawada. Probabilistic latent network visualization: Inferring and embedding diffusion networks. In *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Min.*, 2014, pages 1236–1245.

[17] A. Lancichinetti, S. Fortunato, and F. Radicchi. Benchmark graphs for testing community detection algorithms. *Phys. Rev. E*, 78(4), 2008.

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

[19] A. Lokhov. Reconstructing parameters of spreading models from partial observations. In *Adv. Neural Inf. Process. Syst.*, 2016, pages 3467–3475.

[20] Y. Mehmood, N. Barbieri, F. Bonchi, and A. Ukkonen. CSI: Community-level social influence analysis. In *Proc. Eur. Conf. Mach. Learn. Knowl. Discov. Databases)*, 2013, pages 48–63.

[21] S. Myers and J. Leskovec. On the convexity of latent social network inference. In *Adv. Neural Inf. Process. Syst.*, 2010, pages 1741–1749.

[22] H. Narasimhan, D. C. Parkes, and Y. Singer. Learnability of influence in networks. In *Adv. Neural Inf. Process. Syst.*, 2015, pages 3186–3194.

[23] P. Netrapalli and S. Sanghavi. Learning the graph of epidemic cascades. In *Proc. ACM SIGMETRICS*, 2012, pages 211–222.

[24] M. E. J. Newman. Finding community structure in networks using the eigenvectors of matrices. *Phys. Rev. E*, 74(3):036104, 2006.

[25] R. Nishii. Maximum likelihood principle and model selection when the true model is unspecified. *J. Multivariate Anal.*, 27(2):392–403, 1988.

[26] J. Pouget-Abadie and T. Horel. Inferring graphs from cascades: A sparse recovery framework. In *Proc. Int. Conf. Mach. Learn.*, 2015, pages 977–986.

[27] Y. Rong, Q. Zhu, and H. Cheng. A model-free approach to infer the diffusion network from event cascade. In *Proc. ACM Int. Conf. Inform. Knowl. Manage.*, 2016, pages 1653–1662.

[28] E. Sefer and C. Kingsford. Convex risk minimization to infer networks from probabilistic diffusion data at multiple scales. In *Proc. IEEE Int. Conf. Data Eng.*, 2015, pages 663–674.

[29] Y. Tang, X. Xiao, and Y. Shi. Influence maximization: Near-optimal time complexity meets practical efficiency. In *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2014, pages 75–86.

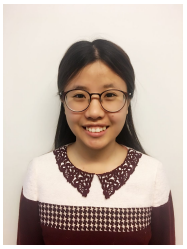
[30] J. Wallinga and P. Teunis. Different epidemic curves for severe acute respiratory syndrome reveal similar impacts of control measures. *Am. J. Epidemiol.*, 160(6):509–516, 2004.

[31] S. Wang, X. Hu, P. Yu, and Z. Li. MMRate: Inferring multi-aspect diffusion networks with multi-pattern cascades. In *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Min.*, 2014, pages 1246–1255.

[32] Q. Yan, H. Huang, Y. Gao, W. Lu, and Q. He. Group-level influence maximization with budget constraint. In *Proc. Int. Conf. Database Syst. Adv. Appl.)*, 2017, pages 625–641.

Hao Huang received the PhD degree in computer science from Zhejiang University, China, in 2012. He is currently an associate professor in the School of Computer Science, Wuhan University, China. His research interests include big data management and analytics, statistical learning, and optimization problems.

Qian Yan received the BS degree in computer science from Wuhan University, China, in 2016. He is currently working toward the MS degree in the School of Computer Science, Wuhan University, China. His research interests include data mining and statistical learning.



Lu Chen received the PhD degree in computer science from Zhejiang University, China, in 2016. She is currently an associate professor at the Department of Computer Science, Aalborg University, Denmark. Her research interests include indexing and querying metric spaces.

Yunjun Gao received the PhD degree in computer science from Zhejiang University, China, in 2008. He is currently a professor in the College of Computer Science, Zhejiang University, China. His research interests include spatial and spatio-temporal databases, metric and incomplete/uncertain data management, and spatio-textual data processing. He is a member of the ACM and the IEEE, and a senior member of the CCF.

Christian S. Jensen is an Obel professor of computer science at Aalborg University, Denmark. His research concerns data management and data-intensive systems, and its focus is on temporal and spatiotemporal data management. He is an editor-in-chief of the *ACM Transactions on Database Systems*, and is a fellow of the ACM and the IEEE.