Zhai *et al. Computational Social Networks* (2015) 2:17
DOI s40649-015-0017-4

Computational Social Networks
a SpringerOpen Journal

RESEARCH Open Access

Cascade source inference in networks: a Markov chain Monte Carlo approach

Xuming Zhai[2], Weili Wu[1,2*] and Wen Xu[2]

*Correspondence:
weiliwu@utdallas.edu
[1]College of Computer Science and Technology, Taiyuan University of Technology, Taiyuan 030024, China
[2]Department of Computer Science, University of Texas at Dallas, 800 W. Campbell Rd, Richardson, TX 75080, USA

Abstract

Cascades of information, ideas, rumors, and viruses spread through networks. Sometimes, it is desirable to find the source of a cascade given a snapshot of it. In this paper, source inference problem is tackled under Independent Cascade (IC) model. First, the #P-completeness of source inference problem is proven. Then, a Markov chain Monte Carlo algorithm is proposed to find a solution. It is worth noting that our algorithm is designed to handle large networks. In addition, the algorithm does not rely on prior knowledge of when the cascade started. Finally, experiments on real social network are conducted to evaluate the performance. Under all experimental settings, our algorithm identified the true source with high probability.

Keywords: Social network; Source inference; Markov chain Monte Carlo

Introduction

Modern social and computer networks are common media for cascades of information, ideas, rumors, and viruses. It is often desirable to identify the source of a cascade from a snapshot of the cascade. For example, a good way to stop a rumor is to find out the person that has fabricated it. Similarly, identifying the first computer infected by a virus provides valuable information for catching the author. Therefore, given the network structure and an observed cascade snapshot consisting only the set of infected/active nodes, solving the source inference problem is very useful in many cases. Hereafter, we use infected/active and infect/activate interchangeably.

In the seminal works [1] and [2], source inference problem under susceptible-infected (SI) model is first studied, and a maximum likelihood estimator is proposed with theoretical performance bound when the network is a tree. Based on the same model, many works solve this problem with different extensions. With a priori knowledge of a candidate source set, reference [3] infers the source node using a maximum a posteriori estimator. Wang et al. [4] utilizes multiple independent epidemic observations to single out their common source. Karamchandani and Franceschetti [5] study the case where infected nodes reveal their infection with a probability. When multiple sources are involved, algorithms are proposed in [6] and [7] to find out all of them. The works mentioned above, except [7], are all based on tree networks, while some of them are applicable to general graphs by constructing breadth-first-search trees. More importantly, all of them use SI model, where an infected node will certainly infect a susceptible neighbor after a random

period of time. Our work, however, is based on Independent Cascade (IC) model. In the IC model, an active node activates its successor with a certain probability determined by the edge weight.

Although SI model is popular in epidemiological researches because it catches the pattern of epidemics, the IC model is arguably more suitable to depict cascades in social networks, where relationship between peers plays a more important role than time of infection. As an example, suppose Alice bought a new hat, her classmates may or may not imitate the purchase depending on how they agree with her taste. Those who do not appreciate her taste are unlikely to change their minds even Alice wears her hat every day. These people are now immune from the influence of Alice's new hat, though they may still be persuaded by someone they appreciate more.

Although the IC model is popular in social network researches, finding source in the IC model is rarely studied. Using a model similar to the IC with identical edge weight, reference[8] studies the problem of inferring both links and sources given multiple observed cascades. Under the IC model, reference [9] solves the problem of finding sources that are expected to generate cascades most similar to the observation. Surprisingly, this problem is fundamentally different from source inference problem, which finds the source that most likely has started the observed cascade. For example, when a cascade that infects all nodes is observed in the simple linear network in Fig. 1, node c is the optimal result for the problem defined in [9] because it is expected to generate a cascade with least difference from the observed one. However, it is obvious that c cannot be responsible for a cascade that spreads through all three nodes.

In this paper, we work on the problem of detecting the source node that is responsible for a given cascade. We first formulate the source inference problem in the IC model and prove its #P-completeness. Then, a Markov chain Monte Carlo (MCMC) algorithm is proposed to solve the inference problem. It is worth noting that our algorithm scales with observed cascade size rather than network size, which is very important due to the huge size of social networks nowadays. Another advantage of our algorithm is that it is designed to deal with snapshot of cascades taken either before or after termination. More importantly, our algorithm does not require prior knowledge of the starting time of the cascade, which is usually unknown in practical scenarios. To evaluate the performance of our algorithm, experiments are done in a real network. Experimental results demonstrate the effectiveness of our algorithm.

Problem formulation

Propagation model

In this work, we model a social network as a weighted directed graph $G(V, E)$ with weights $w_{i,j} \in (0, 1]$ associated with each edge $(i, j) \in E$ representing the probability

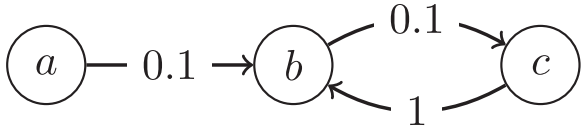


Fig. 1 Example of a simple case of source inference problem: if all three nodes are found active, then node a must be the source

of i successfully influencing j. The propagation procedure of a cascade in the network is depicted by the well-known IC model [10]. The cascade starts with all nodes inactive except a source node s, which we assume is activated at time τ_0. At every time step $\tau > \tau_0$, every node i that was activated at $\tau - 1$ has a single chance to influence each of its inactive successors through the directed edge with success probability specified by the weight of the edge. If the influence is successful, then the successor is activated at time τ and will be able to influence its inactive successors at the next time step. The process terminates when no new node is activated.

An important fact about the IC model is that each active node has only one chance of influencing each of its neighbors. To put it another way, there is only one chance for each edge to participate in the propagation with success rate specified by the weight. Since edge weights are fixed and independent of the cascade, we can flip the biased coins even before the cascade starts to determine whether each edge will help the propagation. This gives an alternative process consisting of two steps that also simulates the IC model. First, a subgraph G' of the original network G is taken by 1) keeping all vertices and 2) filtering edges according to their weights, i.e.,

$$\begin{aligned} \forall v \in G, \quad & v \in G', \\ \forall (i,j) \in G, \quad & \Pr\left((i,j) \in G'\right) = w_{i,j}. \end{aligned} \tag{1}$$

Then, every node i reachable from source s in G' is active, with its activation time set to $\tau + d_{G'}(s,i)$, where $d_{G'}(s,i)$ is the distance, i.e., number of edges in the directed shortest path, from s to i in G'.

It is easy to verify that the alternative process is equivalent to the previous one. Moreover, the alternative view builds the equivalence between sampling subgraphs of network and simulating cascades on it. Due to this convenience, we extensively use the alternative view in the following sections.

Source inference problem

Suppose in a given network G, an unnoticed cascade starts from an unknown source node s^* at time τ_0. Later at time $\tau_0 + \tau$, the cascade is discovered and the set of active nodes A_τ is identified without knowing their corresponding activation time. Note that A_τ can be viewed as a snapshot of the cascade at time τ. Now, we want to find the node $\hat{s}$ that most likely had started the cascade. Thus,

$$\hat{s} = \arg\max_s \Pr\left(A_\tau | G, s, \tau\right), \tag{2}$$

where $\Pr\left(A_\tau | G, s, \tau\right)$ denotes the probability of a cascade on G starting from s having snapshot A_τ at time τ. According to the alternative view of the IC model defined in the 'Propagation model' section and suppose G' is sampled according to (1), we have

$$\Pr\left(A_\tau | G, s, \tau\right) = \Pr\left(A_\tau = \{i \mid d_{G'}(s,i) \leq \tau\}\right). \tag{3}$$

The following theorem shows the intractability of source inference problem, i.e., solving (2) given G, τ, and A_τ.

Theorem 1. *Source inference problem is #P-complete.*

This theorem is proven by constructing a polynomial-time Turing reduction from s-t connectedness problem [11] to source inference problem. Please refer to Appendix 1 for the detailed proof.

Source inference algorithm

Basic algorithm

We use $\mathcal{R}(G', s, \tau)$ to denote the set of nodes in G' reachable from s within distance τ, i.e.,

$$\mathcal{R}(G', s, \tau) = \{i \mid d_{G'}(s, i) \leq \tau\}.$$

Then, the probability shown in (3) can also be written as

$$\Pr(A_\tau | G, s, \tau) = \sum_{G' \subseteq G} \Pr_{\mathcal{G}}(G') I(A_\tau = \mathcal{R}(G', s, \tau)) \tag{4}$$

$$= \mathbb{E}_{G' \sim \mathcal{G}}[I(A_\tau = \mathcal{R}(G', s, \tau))], \tag{5}$$

where $\mathcal{G}$ represents the distribution of subgraphs of G defined by (1), $\Pr_{\mathcal{G}}(G')$ denotes the probability mass function (PMF) of G' in distribution $\mathcal{G}$, i.e.,

$$\Pr_{\mathcal{G}}(G') = \prod_{(i,j) \in G} w_{i,j}^{I((i,j) \in G')} (1 - w_{i,j})^{I((i,j) \notin G')} \tag{6}$$

and I is an indicator function defined as

$$I(c) = \begin{cases} 1 \text{ if condition } c \text{ is true,} \\ 0 \text{ otherwise.} \end{cases}$$

Because of the #P-completeness of source inference problem, calculating exact value of (4) is #P-hard.

A trivial method to approximate the value is to estimate the expectation in (2) by randomly sampling graphs in $\mathcal{G}$. But this method is still impractical. To show this, we define $\mathcal{S} = \{G' \mid G' \subseteq G\}$ as the set of all subgraphs of G, which is also the support of $\mathcal{G}$. Then, a subset of $\mathcal{S}$ is defined as

$$\mathcal{S}' = \{G' \mid G' \subseteq G, \exists s, s \rightsquigarrow A_\tau \subseteq G'\},$$

where $s \rightsquigarrow A_\tau \subseteq G'$ denotes "every node in A_τ is reachable from s in G'". Now, notice that $A_\tau = \mathcal{R}(G', s, \tau) \implies G' \in \mathcal{S}'$ and that the ratio $|\mathcal{S}|/|\mathcal{S}'|$ can be exponential to $|G|$, which means almost all subgraphs of G will make the indicator function in equals 0. As an example, consider a linear graph $G_L(V_L, E_L)$ where $V_L = \{v_1, v_2, \ldots, v_n\}$ and $E_L = \{(v_k, v_{k+1}) \mid 1 \leq k < n\}$. Suppose $A_\tau = V_L$ and $\tau = n$, then $|\mathcal{S}| = 2^{n-1}$ whereas $s \rightsquigarrow A_\tau \subseteq G'$ only if $G'_L = G_L$ and $s = v_1$.

To overcome this problem, we want to sample G' from set $\mathcal{S}'$ rather than $\mathcal{S}$. On set $\mathcal{S}'$, we define a new sampling distribution, denoted as $\mathcal{G}'$, whose PMF is

$$\Pr_{\mathcal{G}'}(G') = \begin{cases} \Pr_{\mathcal{G}'}(G')/Z & \text{if } G' \in \mathcal{S}', \\ 0 & \text{otherwise,} \end{cases}$$
$$Z = \sum_{G' \in \mathcal{S}'} \Pr_{\mathcal{G}'}(G'). \tag{7}$$

Notice that set $\mathcal{S}'$ is independent of any candidate source node, so is the normalization factor Z. Therefore, with (7), we have

$$\mathbb{E}_{G' \sim \mathcal{G}}[I(A_\tau = \mathcal{R}(G', s, \tau))] \propto \mathbb{E}_{G' \sim \mathcal{G}'}[I(A_\tau = \mathcal{R}(G', s, \tau))].$$

Consequently, we can solve source inference problem (2) by solving

$$\hat{s} = \arg\max_{s} \mathbb{E}_{G' \sim \mathcal{G}'}[\, I(A_\tau = \mathcal{R}(G', s, \tau))] \,. \tag{8}$$

Now the problem is how to sample from $\mathcal{S}'$ with probability defined in (7). However, one can easily show that calculating factor Z is #P-hard, which makes calculating (7) impractical. Therefore, it is unlikely to be possible to directly sample from set $\mathcal{S}'$. Fortunately, the probability ratio between any two subgraphs is easy to compute; thus, we can use Metropolis algorithm to sample distribution $\mathcal{G}'$ in a Markov chain Monte Carlo.

Algorithm 1: Local move

Input: G'_k, G, $\boldsymbol{w}$, A_τ

Output: G'_{k+1}

choose $(i, j) \in G$ uniformly randomly;
if $(i, j) \in G'_k$ **then** // remove edge
  $G'_{k+1} = G'_k \setminus \{(i, j)\}$;
  $p = (1 - w_{i,j})/w_{i,j}$;
else // add edge
  $G'_{k+1} = G'_k \cup \{(i, j)\}$;
  $p = w_{i,j}/(1 - w_{i,j})$;
if $\exists i, i \rightsquigarrow A_\tau \subseteq G'_{k+1}$ **then**
  $p = \min\{p, 1\}$;
else
  $p = 0$;
with probability $1 - p$, set $G'_{k+1} = G'_k$; // reject
return G'_{k+1};

Algorithm 1 describes a local move from a subgraph in $\mathcal{S}'$ to another. Each local move will add/remove an edge to/from the previous subgraph G'_k. The new subgraph G'_{k+1} is either accepted or rejected depending on the probability ratio $\Pr_{\mathcal{G}'}(G'_{k+1})/\Pr_{\mathcal{G}'}(G'_k)$ defined in $\mathcal{G}'$. Starting from any subgraph in $\mathcal{S}'$, running Algorithm 1 iteratively will produce a Markov chain whose states represent subgraphs in $\mathcal{S}'$ and whose stationary distribution is exactly the same as (7).

With the help of local move in Algorithm 1, Algorithm 2 infers the most likely source node responsible for the cascade snapshot A_τ taken at time τ. Input parameter K is used to indicate the number of samples to take by this algorithm. With line 3, the algorithm starts with whole graph G as the initial sample, which is obviously in $\mathcal{S}'$. During every iteration of the while-loop, a subgraph in $\mathcal{S}'$ is sampled, and all possible source vertices are found and recorded. After the while-loop ends, $count[\, i] \,/K$ is the estimation of $\mathbb{E}_{G' \sim \mathcal{G}'}[\, I(A_\tau = \mathcal{R}(G', i, \tau))]$. Hence, the returned value of Algorithm 2 is an approximate solution of (8).

A more practical approach

Algorithm 2 has some drawbacks in practical scenarios. First, the whole network may be orders of magnitude larger than the cascade snapshot in question. However, Algorithm 2

Algorithm 2: Basic source inference algorithm

Input: instance: G, $\boldsymbol{w}$, A_τ, τ; parameter: K

Output: s

create new array *count* with size $|V|$ and default value 0;
$k = 0$;
$G'_k = G$; // initial sample in $\mathcal{S}'$
while $k < K$ **do**
  $\mathcal{C} = \{i \mid \mathcal{R}(G'_k, i, \tau) = A_\tau\}$;
  for $i \in \mathcal{C}$ **do**
    if $A_\tau == \mathcal{R}(G'_k, i, \tau)$ **then**
      $count[i] = count[i] + 1$;
  run Algorithm 1 with G'_k G, $\boldsymbol{w}$, A_τ to get G'_{k+1};
  $k = k + 1$;
$s = \arg\max_i count[i]$;
return s;

scales with the size of full network rather than the snapshot, which is unfavorable here. Second, when the source node of a cascade is unknown, the starting time of the cascade is usually also absent. In these cases, inferring source node without knowing τ is desired. In this section, we will handle these two problems.

Based on the cascade snapshot A_τ, we can classify edges in E into three disjoint subsets

$$\begin{aligned} E_1 &= \{(i,j) \mid (i,j) \in E, i,j \in A_\tau\}, \\ E_2 &= \{(i,j) \mid (i,j) \in E, i \in A_\tau, j \notin A_\tau\}, \\ E_3 &= \{(i,j) \mid (i,j) \in E, i \notin A_\tau\}. \end{aligned} \tag{9}$$

And E_2 can be further split into subsets according to the source node of edges:

$$E_{2,u} = \{(i,j) \mid (i,j) \in E_2, i = u\}.$$

Then we define three subgraphs of $G(V,E)$ accordingly: $G_1(A_\tau, E_1)$, $G_2(V,E_2)$, and $G_3(V,E_3)$. Note that G_1 only contains nodes in A_τ because edges in G_1 are all between nodes in A_τ. Furthermore, we partition each sampled subgraph G' into G'_1, G'_2 and G'_3, where $G'_k = G' \cap G_k$. With these definitions, we have the following lemma.

Lemma 1. *If we define subgraph $G_1(A_\tau, E_1)$ consisting of only edges between nodes in A_τ, the condition*

$$A_\tau = \mathcal{R}(G', s, \tau) \tag{10}$$

is equivalent to the combination of

$$A_\tau = \mathcal{R}(G'_1, s, \tau) \tag{11}$$

and

$$\forall i \in A_\tau, \quad d_{G'_1}(s,i) = \tau \ \vee \ E_{2,i} \cap E' = \varnothing, \tag{12}$$

where $G'_1 = G' \cap G_1$.

Proof. Eq. 10 can be split to 1) any node in A_τ must be within distance τ from s, i.e.,

$$A_\tau \subseteq \mathcal{R}(G', s, \tau), \tag{13}$$

and 2) any node outside A_τ must have distance from s larger than τ, i.e.,

$$\mathcal{R}(G', s, \tau) \setminus A_\tau = \varnothing. \tag{14}$$

Hence, the shortest path from s to any node $i \in A_\tau$ is within G_1, which implies $\forall i \in A_\tau$, $d_{G'}(s,i) = d_{G'_1}(s,i)$ and thus (11). Further, (12) means any node i with $d_{G'}(s,i) < \tau$ must not be able to activate its neighbors outside A_τ, which is necessary to ensure (14).

On the other hand, (11) guarantees (13) and (12) ensures $\forall i \notin A_\tau, d_{G'}(s,i) > \tau$ which leads to (14). □

From Lemma 1, it is straightforward to get the following corollaries.

Corollary 1. *The indicator function in (4) is equivalent to*

$$\begin{aligned} I(A_\tau = \mathcal{R}(G', s, \tau)) =& I(A_\tau = \mathcal{R}(G'_1, s, \tau)) \\ & \cdot \prod_{(i,j)\in G'_2} I(d_{G'_1}(s,i) = \tau). \end{aligned}$$

Corollary 2. $I(A_\tau = \mathcal{R}(G', s, \tau))$ *is independent of* G'_3.

In addition, because $G' = G'_1 \cup G'_2 \cup G'_3$ and edge sets in G'_k are disjoint, (6) can be rewritten as the product of three terms

$$\begin{aligned} \Pr_{\mathcal{G}}(G') &= \prod_{(i,j)\in G} w_{i,j}^{I((i,j)\in G')} (1 - w_{i,j})^{I((i,j)\notin G')} \\ &= \prod_{k=1}^{3} \Pr_{\mathcal{G}k}(G'_k), \end{aligned} \tag{15}$$

where

$$\Pr_{\mathcal{G}_k}(G'_k) = \prod_{(i,j)\in G_k} w_{i,j}^{I((i,j)\in G'_k)} (1 - w_{i,j})^{I((i,j)\notin G'_k)}. \tag{16}$$

Now we have Theorem 2 that speedup the algorithm.

Theorem 2. *Define distribution* $\mathcal{G}'_1$ *of graphs in* $\mathcal{S}'_1 = \{G'_1 \mid G'_1 \subseteq G_1, \exists s, s \rightsquigarrow A_\tau \subseteq G'_1\}$ *with PMF proportional to* $Pr_{\mathcal{G}_1}(G'_1)$. *Then, we have*

$$\Pr(A_\tau | G, s, \tau) \propto \mathbb{E}_{G'_1 \sim \mathcal{G}'_1}[f(G'_1, s, \tau)], \tag{17}$$

where

$$f(G'_1, s, \tau) = I(A_\tau = \mathcal{R}(G'_1, s, \tau)) \prod_{\substack{(i,j)\in G_2 \\ d_{G'_1}(s,i)<\tau}} (1 - w_{i,j}). \tag{18}$$

The proof of Theorem 2 is shown in Appendix 2.

Theorem 2 shows that sampling subgraphs of G_1, rather than the whole network G, is sufficient to infer the cascade source, which greatly accelerates the algorithm when the whole network is much larger than the cascade snapshot A_τ.

Next, we deal with unknown cascade starting time, i.e., unknown τ. First, due to the fact that node set in G_1 is A_τ,

$$\begin{aligned} A_\tau = \mathcal{R}(G_1', s, \tau) &\Longleftrightarrow A_\tau \subseteq \mathcal{R}(G_1', s, \tau) \\ &\Longleftrightarrow s \rightsquigarrow A_\tau \subseteq G_1' \wedge \tau \geq \epsilon_{G_1'}(s), \end{aligned}$$

where $\epsilon_{G_1'}(s)$ is the eccentricity of node s in G_1', defined as

$$\epsilon_{G_1'}(s) = \max_{i \in G_1'} d_{G_1'}(s, i). \tag{19}$$

As a result, for any given G_1' and s such that $s \rightsquigarrow A_\tau \subseteq G_1'$, there are three possible values for function $f(G', s, \tau)$ in (18):

$$f(G', s, \tau) = \begin{cases} 0, & \tau < \epsilon_{G_1'}(s), \\ \prod\limits_{\substack{(i,j) \in G_2 \\ d_{G_1'}(s,i) < \epsilon_{G_1'}(s)}} (1 - w_{i,j}), & \tau = \epsilon_{G_1'}(s), \\ \prod\limits_{(i,j) \in G_2} (1 - w_{i,j}), & \tau > \epsilon_{G_1'}(s). \end{cases} \tag{20}$$

Here, the values for all three cases are independent of τ. Then, we have Theorem 3 that deals with unknown cascade starting time.

Theorem 3. *Suppose samples $G_{1,k}'$, $k = 1, 2, \ldots, K$ are taken with distribution $\mathcal{G}_1'$, then we can approximate (17) by*

$$\mathbb{E}_{G_1' \sim \mathcal{G}_1'}[f(G_1', s, \tau)] \approx \frac{1}{K} \left(A(s, \tau) + W \cdot \sum_{\tau' < \tau} C(s, \tau') \right),$$

where

$$\begin{aligned} A(s, \tau) &= \sum_{k: \tau = \epsilon_{G_{1,k}'}(s)} \prod_{\substack{(i,j) \in G_2 \\ d_{G_{1,k}'}(s,i) < \epsilon_{G_{1,k}'}(s)}} (1 - w_{i,j}), \\ W &= \prod_{(i,j) \in G_2} (1 - w_{i,j}), \\ C(s, \tau') &= \sum_{k: \tau' = \epsilon_{G_{1,k}'}(s)} 1. \end{aligned} \tag{21}$$

Proof. Because samples $G_{1,k}'$ are taken with distribution $\mathcal{G}_1'$, we have

$$\mathbb{E}_{G_1' \sim \mathcal{G}_1'}[f(G_1', s, \tau)] \approx \frac{1}{K} \sum_{k=1}^{K} f(G_{1,k}', s, \tau). \tag{22}$$

Substituting (20) into the summation of (22) proves the theorem. □

With both Theorems 2 and 3, Algorithm 2 can be improved to Algorithm 3 which overcomes problems of large network and unknown τ.

In Algorithm 3, we only consider τ' ranging from 1 to $|A_\tau|$ because 1) $\epsilon_{G_{1,k}'}(s)$ ranges from 1 to $|A_\tau| - 1$ given $|A_\tau| > 1$ and $s \rightsquigarrow A_\tau \subseteq G_{1,k}'$; 2) if $\tau >= |A_\tau|$, the cascade must have terminated, thus $\forall \tau > |A_\tau|, \Pr(A_\tau | G, s, \tau) = \Pr(A_\tau | G, s, |A_\tau|)$. The input time range $[\tau_l, \tau_u]$ represents limited knowledge of τ. If the exact starting time of the cascade

Algorithm 3: Advanced source inference algorithm

Input: instance: G, $\boldsymbol{w}$, A_τ; time range $[\tau_l, \tau_u]$; parameter: K

Output: s

1 create new tables *accu*, *count* and *result* with size $|A_\tau| \times |A_\tau|$ and default value 0;
2 create graph $G_1(A_\tau, E_1)$ according to (9);
3 $k = 0$;
4 $G'_{1,k} = G'_1$;
5 calculate W by (21);
6 **for** $i \in A_\tau$ **do**
7 $\quad W_i = \prod_{j \in V \setminus A_\tau : (i,j) \in G} (1 - w_{i,j})$;
8 **while** $k < K$ **do**
9 $\quad \mathcal{C} = \{i \mid i \rightsquigarrow A_\tau \subseteq G'_{1,k}\}$;
10 $\quad$ **for** $i \in \mathcal{C}$ **do**
11 $\quad\quad \tau' = \epsilon_{G'_{1,k}}(i)$ by (19);
12 $\quad\quad w' = W$;
13 $\quad\quad$ **for** $j \in A_\tau$, $d_{G'_{1,k}}(i,j) == \tau'$ **do**
14 $\quad\quad\quad w' = w'/W_j$;
15 $\quad\quad accu[\,i\,]\,[\,\tau'\,] = accu[\,i\,]\,[\,\tau'\,] + w'$;
16 $\quad\quad count[\,i\,]\,[\,\tau'\,] = count[\,i\,]\,[\,\tau'\,] + 1$;
17 $\quad$ run Algorithm 1 with $G'_{1,k}$ G_1, $\boldsymbol{w}$, A_τ to get $G'_{1,k+1}$;
18 $\quad k = k + 1$;
19 **for** $i \in A_\tau$ **do**
20 $\quad c = 0$;
21 $\quad$ **for** $\tau' = 1, \ldots, |A_\tau|$ **do**
22 $\quad\quad result[\,i\,]\,[\,\tau'\,] = accu[\,i\,]\,[\,\tau'\,] + W \times c$;
23 $\quad\quad c = c + count[\,i\,]\,[\,\tau'\,]$;
24 $s = \arg\max_i \sum_{\tau'=\tau_l}^{\tau_u} result[\,i\,]\,[\,\tau'\,]$;
25 **return** s;

is known, we can use $\tau_l = \tau_u = \tau$. On the contrary, if nothing at all is known about τ, $\tau_l = \min_i \epsilon_{G'_1}(i)$ and $\tau_u = |A_\tau|$ may be used instead.

It should be noted that for any sample G'_1, line 9 in Algorithm 3 can be done in $O(|G'_1|)$ time. First, condensation $C(G'_1)$ is calculated, which needs linear time. Then, since $C(G'_1)$ is a directed acyclic graph, there is at least one strong component in $C(G'_1)$ that has no predecessor. If there is exactly one such component, it is the set $\mathcal{C}$; if there is more than one, $\mathcal{C} = \varnothing$. This method also applies to line 8 in Algorithm 1 and line 5 in Algorithm 2.

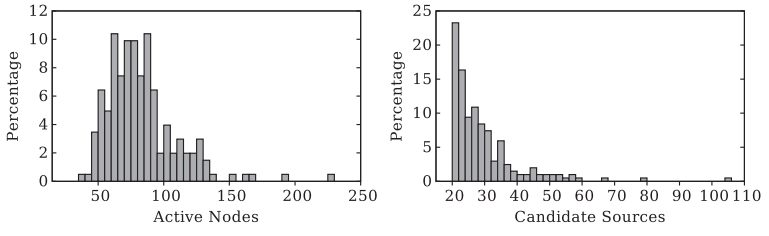
Experimental results

In this section, we conduct experiments of our cascade source inference algorithm (Algorithm 3, with $K = 10^6$) on real network dataset. The network used is from WikiVote dataset ([12, 13]), which consists of all Wikipedia voting data from the inception of Wikipedia till January 2008. The dataset has 7115 nodes and 103,689 directed unweighted

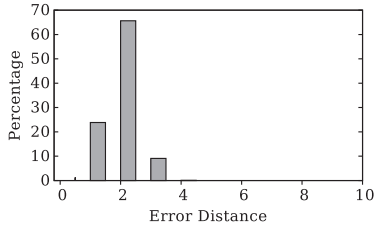
edges. Each node represents a Wikipedia user participating in elections, while each directed node (i, j) means user i voted for user j. We use this unweighted dataset because we cannot find a social network dataset with influence probability available despite our best effort. Since the dataset is unweighted, we use reciprocal of in-degree of the destination node as the weight of an edge. With uniformly randomly chosen source nodes, cascades are then generated on the network according to the IC model. To make the experiment challenging, we discard cascades with less than 20 candidate sources. Here, candidate source set is not active nodes set A_τ, but set of nodes from which all active nodes are reachable in G_1, i.e., $\{i \mid i \rightsquigarrow A_\tau \subseteq G_1\}$. We use 200 cascades in our experiments. Figure 2a, b shows histograms of the number of active nodes and candidate sources among these cascades.

To compare our proposed algorithm with existing algorithm, we also implement the algorithm proposed by [9]. In that paper, they proposed three algorithms ("DP", "Sort", and "OutDegree") to find a set of k sources. In our case where single source generates the cascade, their DP algorithm and Sort' algorithm are equivalent. In the experiment below, we use this algorithm and call it "Effector" algorithm.

First, we take snapshot at $\tau = |A_\tau|$, i.e., after cascades terminate and do the experiment with exact knowledge of τ. Figure 3a shows the distribution of error distances, which is defined as the distance between inferred source node and true source node assuming edges are undirected. To compare with, the error distance of random guess among A_τ is also shown in Fig. 1c. It is clear that all source nodes inferred by our algorithm are within two hops around the source node, and 24 % of the inferred nodes are true sources. In comparison, the Effector algorithm has fewer results with 0 or 1 error distance. To further evaluate the algorithm, we make the algorithms output a list of candidate source nodes sorted in descending order of likelihood, rather than merely the most likely source node. This output is sometimes more useful because it answers queries like "what's the 5 most

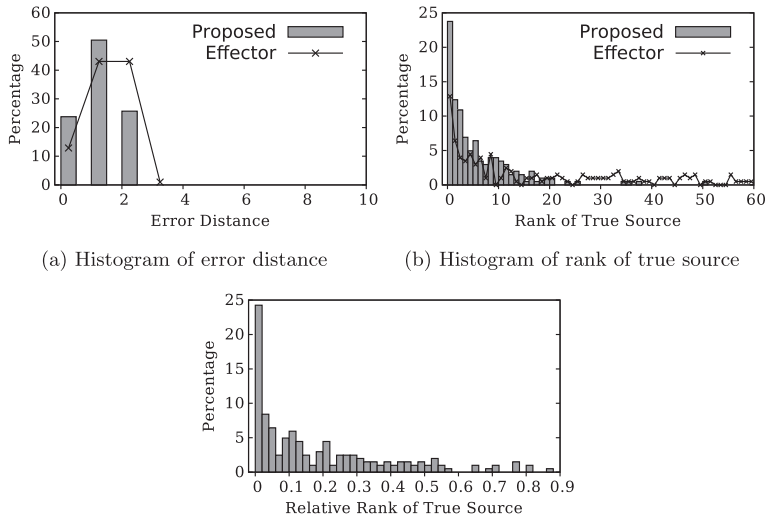


(a) Histogram of number of active nodes (b) Histogram of number of candidate sources



(c) Histogram of error distance by random guess

Fig. 2 Statistics of cascades used in experiments. **a** Histogram of number of active nodes. **b** Histogram of number of candidate sources. **c** Histogram of error distance by random guess



(a) Histogram of error distance

(b) Histogram of rank of true source

(c) Histogram of relative rank of true Source

Fig. 3 Experimental result: $\tau = |A_\tau|$, τ known. **a** Histogram of error distance. **b** Histogram of rank of true source. **c** Histogram of relative rank of true source

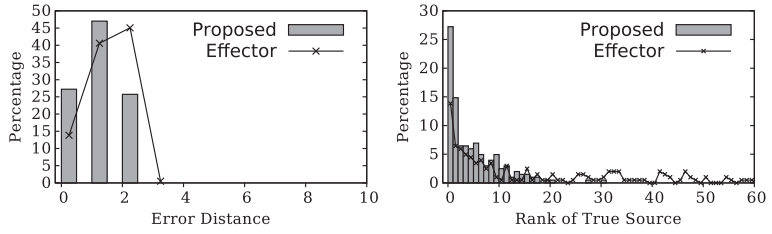
likely source of the cascade". Figure 3b shows the distribution of rank of the true source node in the ordered list. In more than half of total experiments, the true source is among top 4 candidates output by our algorithm. The Effector algorithm, however, has a much heavier tail with far less results with lower ranks. In fact, there are 15 % of the results with a rank higher than 60 which is not shown in the figure. Figure 3c shows distribution of relative ranks, i.e., rank divided by candidate set size. Only our algorithm is shown in this figure because the Effector algorithm does not calculate candidate set and their output list include many nodes not in the candidate set due to the reason explained by Fig. 1 in the 'Introduction' section. In more than 50 % of the experiments, our output that has relative rank of the true source is less than or equal to 0.1.

Then, we do experiments with snapshots taken at $\tau = 8$, when most of the cascades are yet to terminate. The results are shown in Fig. 4. Similarly, our proposed algorithm performs better than the Effector algorithm. In 55 % of the experiments, our algorithm has true source node among top 4 candidates, and in half of experiments, we have true source node with relative rank no larger than 0.1.

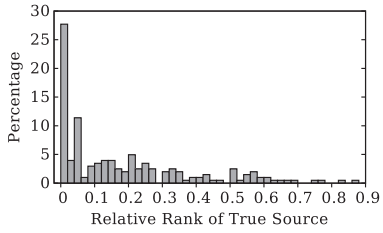
To evaluate the performance of our source inference algorithm when exact cascade starting time is absent, we conduct another experiment on the snapshot taken at $\tau = 8$ with input time range $[0, 16]$. As shown in Fig. 5, our algorithm effectively infers the source nodes even without exact knowledge of cascade starting time. In the experiment, 57 % of the true source nodes are among top 4 candidates, and in half of the cases, the true source ranked top 10 % in the output list.

Conclusion

We considered cascade source inference problem in the IC model. First the #P-completeness of this problem was proven. Then, a Markov chain Monte Carlo algorithm was proposed to approximate the solution. Our algorithm was designed with two major



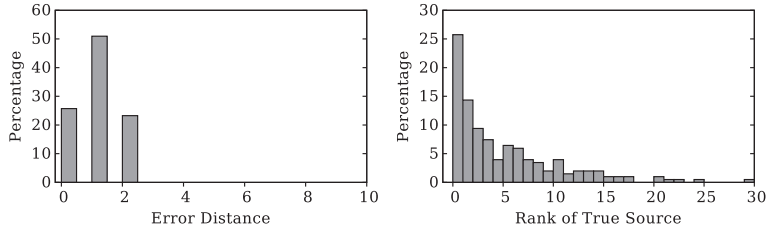
(a) Histogram of error distance (b) Histogram of rank of true source



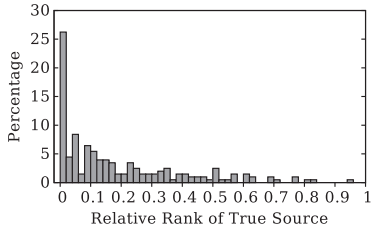
(c) Histogram of relative rank of true Source

Fig. 4 Experimental result: $\tau = 8$, τ known. **a** Histogram of error distance. **b** Histogram of rank of true source. **c** Histogram of relative rank of true source

advantages: 1) it scales with the observed cascade snapshot rather than the whole network and thus is applicable to enormous modern social networks and 2) it does not require any knowledge about the starting time of the cascade, which is a common and practical scenario in cascade source inference problem. To demonstrate the performance of our algorithm, experiments on real social network were conducted. As shown above, our algorithm performs well no matter when the cascade snapshot is taken or whether the cascade starting time is known. In all these experiments, around 25 % of the true sources are correctly identified, about half of them are among the top 4 or top 10 % of the candidates.



(a) Histogram of error distance (b) Histogram of rank of true source



(c) Histogram of relative rank of true Source

Fig. 5 Experimental result: $\tau = 8$, τ unknown, $\tau_l = 0$, $\tau_u = 16$. **a** Histogram of error distance. **b** Histogram of rank of true source. **c** Histogram of relative rank of true source

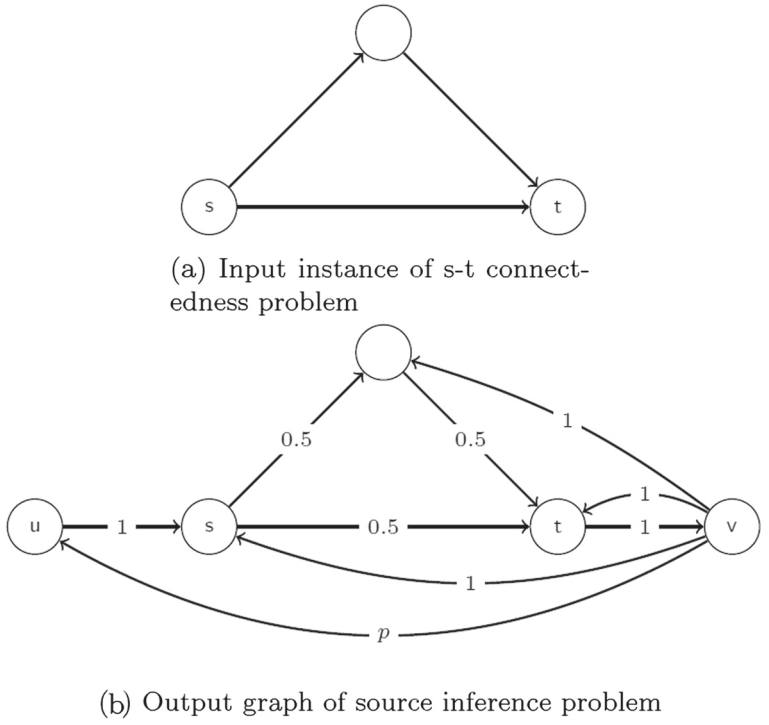


Fig. 6 Example of Algorithm 4. **a** Input instance of s-t connectedness problem. **b** Output graph of source inference problem

Appendix 1

Proof of Theorem 1

We will prove Theorem 1 by constructing a polynomial-time Turing reduction from s-t connectedness problem to source inference problem. S-t connectedness problem is given a directed graph $\hat{G}(\hat{V},\hat{E})$ and two nodes $s,t \in \hat{V}$, output the number of subgraphs of $\hat{G}$ in which there is a path from s to t, i.e. , $\text{Connectedness}(\hat{G},s,t) = |\{\hat{E}' \subseteq \hat{E} \mid |s \rightsquigarrow t \subseteq \hat{E}'\}$. This problem is known to be #P-complete [11].

A key part of the proof is Algorithm 4 which converts an instance of s-t connectedness problem to an instance of source inference problem with properties listed in Lemma 2. An simple example of this algorithm is shown in Fig. 6.

Algorithm 4: Conversion from an instance of s-t connectedness problem to an instance of source inference problem

Input: Parameter $p \in (0,1]$; instance $\hat{G}(\hat{V},\hat{E}), s,t \in \hat{V}$.

Output: $G(V,E)$, $\boldsymbol{w}$, A_τ, τ.

$V = \hat{V} \cup \{u,v\}, u,v \notin \hat{V}$;

$E = \hat{E} \cup \{(v,u),(u,s),(t,v)\} \cup \{(v,j) \mid \forall j \in \hat{V}\}$;

$w_{v,u} = p, w_{u,s} = 1, w_{t,v} = 1$;

$w_{v,j} = 1, \forall j \in \hat{V}$;

$w_{i,j} = 0.5, \forall (i,j) \in \hat{E}$;

$A_\tau = V$;

$\tau = |V|$;

return $G(V,E)$, $\boldsymbol{w}$, A_τ, τ;

Lemma 2. *Given input parameter p and instance* $\hat{G}(\hat{V},\hat{E})$, $s,t \in \hat{V}$, *the output instance* $G(V,E)$, $\boldsymbol{w}$, A_τ, τ *of Algorithm 4 has the following properties:*

1. $\Pr(A_t|G,v,\tau) = \Pr(A_t|G,t,\tau) = p$;
2. $\Pr(A_t|G,i,\tau < p, \forall i \in \hat{V}, i \neq t$;
3. $\Pr(A_t|G,u,\tau) = \text{Connectedness}(\hat{G},s,t) \cdot 0.5^{|\hat{E}|}$.

Proof. In this proof, we use $i \rightsquigarrow j \subseteq G$ to denote the existence of a path from i to j in graph G. In addition, $i \rightsquigarrow V \subseteq G$ means $\forall j \in V, j \neq i, i \rightsquigarrow j \subseteq G$.

According to the algorithm, output snapshot A_τ contains all vertices, and $\tau = |V|$ guarantees that $d_{G'}(i,j) < \tau$ if $i \rightsquigarrow j \subseteq G'$. Therefore, due to (3), the output instance has

$$\Pr(A_\tau|G,i,\tau = \Pr(i \rightsquigarrow V \subseteq G'),$$

which means considering reachability rather than distance is sufficient in the remaining part of the proof.

Now, due to line 4 in Algorithm 4, every node in $\hat{V}$ is reachable from v in every subgraph G' sampled via (1). And because $w_{t,v} = 1$ (by line 3), for any subgraph G',

$$\begin{aligned} &t \rightsquigarrow \hat{V} \subseteq G', \\ &v \rightsquigarrow \hat{V} \subseteq G', \\ &\forall i \in V, \quad i \rightsquigarrow t \subseteq G' \iff i \rightsquigarrow v \subseteq G' \iff i \rightsquigarrow \hat{V} \subseteq G'. \end{aligned} \tag{23}$$

Thus property 1 is straightforward:

$$\begin{aligned} \Pr(A_t|G,t,\tau) &= \Pr(A_t|G,v,\tau) \\ &= \Pr(v \rightsquigarrow V \subseteq G') \\ &= \Pr(v \rightsquigarrow u \subseteq G') \\ &= p. \end{aligned}$$

On the other hand, since the new node u has only one incoming edge (v,u), we have $\forall i \in \hat{V}, i \neq t, i \rightsquigarrow u \subseteq G'$ implies $i \rightsquigarrow t \subseteq G'$. Therefore, we have the proof for property 2: for any $i \in \hat{V}, i \neq t$,

$$\begin{aligned} \Pr(A_t|G,i,\tau) &= \Pr(i \rightsquigarrow V \subseteq G') \\ &= \Pr(i \rightsquigarrow t \subseteq G') \cdot p \\ &< p, \end{aligned}$$

where the last inequality is because every incoming edge of t has weight 0.5 according to line 5 in Algorithm 4.

To prove property 3, we first note that s is the only successor of u and $w_{u,s} = 1$, with (23), we have

$$u \rightsquigarrow V \subseteq G' \iff s \rightsquigarrow t \subseteq G'.$$

And therefore,

$$\Pr(A_t|G,u,\tau) = \Pr(u \rightsquigarrow V \subseteq G') = \Pr(s \rightsquigarrow t \subseteq G'). \tag{24}$$

Because $\hat{G} \subset G$, sampling subgraphs G' of G can be viewed as sampling subsets of $\hat{E}$ followed by sampling subsets of $E \setminus \hat{E}$. Since any path from s to t consists only edges in $\hat{E}$, $\Pr(s \rightsquigarrow t \subseteq G')$ is fully determined by sampling $\hat{E}$, or equivalently, sampling subgraphs of $\hat{G}$. As a result,

$$\Pr(s \rightsquigarrow t \subseteq G') = \text{Connectedness}(\hat{G}, s, t) \cdot 0.5^{|\hat{E}|}, \tag{25}$$

because every subset of $\hat{E}$ has probability $0.5^{|\hat{E}|}$ to be selected via (1) according to line 5 in Algorithm 4. Now property 3 follows from (24) and (25). □

Proof. First, to show source inference problem is in #P, we note that calculating $\Pr(A_t|G, i, \tau)$ is in #P since it is the sum of probabilities of all subgraphs of G with $i \rightsquigarrow V \subseteq G$. So source inference problem, i.e., finding node i that maximize $\Pr(A_t|G, i, \tau)$, is also in #P.

Since graph $\hat{G}$ has $2^{|\hat{E}|}$ subgraphs, Connectedness$\hat{G}, s, t$ must be an integer in range $[0, 2^{|\hat{E}|}]$. Therefore, $\Pr(A_t|G, u, \tau)$ of the output instance of Algorithm 4 must be in set $\{k \cdot 0.5^{|\hat{E}|} \mid k \in \mathbb{N}, k \leq 2^{|\hat{E}|}\}$. A binary search algorithm, i.e., Algorithm 5, can solve s-t connectedness problem by solving source inference problem.

Algorithm 5: Solution of s-t connectedness problem with oracle for source inference problem

Input: $\hat{G}(\hat{V}, \hat{E}), s, t \in \hat{V}$.

Output: $k =$ Connectedness$\hat{G}, s, t$

```
1  if t is not reachable from s then
2      return 0;
3  m = 2^|Ê|, n = 1;
4  while m ≠ n do
5      p = (m + n)/2 · 0.5^|Ê|;
6      run Algorithm 4 with p, Ĝ, s, t to get G, w, A_τ, τ;
7      solve source inference problem G, w, A_τ, τ to get x;
8      if x == u then
9          n = (m + n + 1)/2;
10     else
11         m = (m + n − 1)/2;
12 return m;
```

In Algorithm 5, there will be $|\hat{E}|$ iterations of while-loop. Hence, only polynomial number of queries to the oracle will be made. All other operations can be done in polynomial time. Therefore, this algorithm shows a polynomial-time Turing reduction from s-t connectedness problem to source inference problem. Since s-t connectedness problem is #P-complete and source inference problem is in #P, Theorem 1 is proven. □

Appendix 2

Proof of Theorem 2

The proof is shown from Eqs. (26) to (33). Here, Eq. (27) follows from (15); (28) is due to the equivalence between sampling $G' \subseteq G$ and sampling $G'_k \subseteq G_k, k = 1, 2, 3$, separately;

(29) results from Corollary 2 and the fact that $\Pr_{\mathcal{G}_k}(G'_k)$ depends only on G'_k respectively; (30) is simply due to $\sum_{G'_3 \subseteq G_3} \Pr_{\mathcal{G}_3}(G'_3) = 1$; (31) is by Corollary 1.

To further transform the (32), we split G_2 to $G_{2,\tau}(V, E_{2,\tau})$ and $G_{2,\hat{\tau}}(V, E_{2,\hat{\tau}})$, where

$$E_{2,\tau} = \bigcup_{\substack{i \in A_\tau \\ d_{G'_1}(s,i) = \tau}} E_{2,i},$$

$$E_{2,\hat{\tau}} = \bigcup_{\substack{i \in A_\tau \\ d_{G'_1}(s,i) < \tau}} E_{2,i}.$$

Then, with given subgraph $G'_1 \subseteq G_1$, sampling subgraph $G'_2 \subseteq G_2$ is essentially sampling $G'_{2,\tau} \subseteq G_{2,\tau}$ and $G'_{2,\hat{\tau}} \subseteq G_{2,\hat{\tau}}$, which leads to (34). Since the first summation in (34) is the sum probability of all possible subgraphs of $G_{2,\tau}$, which is 1, we have (35). Because only one specific subgraph $G'_{2,\hat{\tau}} \subseteq G_{2,\hat{\tau}}$, namely, $G'_{2,\hat{\tau}} = G_{2,\hat{\tau}}$, satisfies $\forall (i,j) \in G_{2,\hat{\tau}}$, $I((i,j) \notin G'_{2,\hat{\tau}}) > 0$, we have (36). Then, substituting (36) into (31) gives (32). According to the definition of distribution $\mathcal{G}'_1$, we have (33) and prove Theorem 2.

$$\Pr(A_\tau | G, s, \tau) = \sum_{G' \subseteq G} \Pr_{\mathcal{G}}(G') I(A_\tau = \mathcal{R}(G', s, \tau)) \tag{26}$$

$$= \sum_{G' \subseteq G} \prod_{k=1}^{3} \Pr_{\mathcal{G}_k}(G'_k) I(A_\tau = \mathcal{R}(G', s, \tau)) \tag{27}$$

$$= \sum_{G'_1 \subseteq G_1} \sum_{G'_2 \subseteq G_2} \sum_{G'_3 \subseteq G_3} \prod_{k=1}^{3} \Pr_{\mathcal{G}_k}(G'_k) I(A_\tau = \mathcal{R}(G', s, \tau)) \tag{28}$$

$$= \sum_{G'_1 \subseteq G_1} \left[\Pr_{\mathcal{G}_1}(G'_1) \cdot \sum_{G'_2 \subseteq G_2} \left[\Pr_{\mathcal{G}_2}(G'_2) I(A_\tau = \mathcal{R}(G', s, \tau)) \cdot \sum_{G'_3 \subseteq G_3} \left[\Pr_{\mathcal{G}_3}(G'_3) \right] \right] \right] \tag{29}$$

$$= \sum_{G'_1 \subseteq G_1} \left[\Pr_{\mathcal{G}_1}(G'_1) \cdot \sum_{G'_2 \subseteq G_2} \left[\Pr_{\mathcal{G}_2}(G'_2) I(A_\tau = \mathcal{R}(G', s, \tau)) \right] \right] \tag{30}$$

$$= \sum_{G'_1 \subseteq G_1} \left[\Pr_{\mathcal{G}_1}(G'_1) I(A_\tau = \mathcal{R}(G'_1, s, \tau)) \cdot \sum_{G'_2 \subseteq G_2} \left[\Pr_{\mathcal{G}_2}(G'_2) \prod_{(i,j) \in G'_2} I(d_{G'_1}(s,i) = \tau) \right] \right] \tag{31}$$

$$= \sum_{G'_1 \subseteq G_1} \left[\Pr_{\mathcal{G}_1}(G'_1) I(A_\tau = \mathcal{R}(G'_1, s, \tau)) \cdot \prod_{\substack{(i,j) \in G_2 \\ d_{G'_1}(s,i) < \tau}} (1 - w_{i,j}) \right] \tag{32}$$

$$\propto \mathbb{E}_{G'_1 \sim \mathcal{G}'_1} \left[I(A_\tau = \mathcal{R}(G'_1, s, \tau)) \cdot \prod_{\substack{(i,j) \in G_2 \\ d_{G'_1}(s,i) < \tau}} (1 - w_{i,j}) \right], \tag{33}$$

where (32) is due to

$$\sum_{G_2' \subseteq G_2} \left[\Pr_{\mathcal{G}_2}(G_2') \prod_{(i,j)\in G_2'} I(d_{G_1'}(s,i) = \tau) \right]$$

$$= \sum_{G_2' \subseteq G_2} \left[\prod_{(i,j)\in G_2} w_{i,j}^{I((i,j)\in G_2')} (1-w_{i,j})^{I((i,j)\notin G_2')} \cdot \prod_{\substack{(i,j)\in G_2 \\ d_{G_1'}(s,i)<\tau}} I((i,j) \notin G_2') \right] \quad \text{(by (16))}$$

$$= \sum_{G_2' \subseteq G_2} \left[\prod_{\substack{(i,j)\in G_2 \\ d_{G_1'}(s,i)=\tau}} w_{i,j}^{I((i,j)\in G_2')} (1-w_{i,j})^{I((i,j)\notin G_2')} \cdot \prod_{\substack{(i,j)\in G_2 \\ d_{G_1'}(s,i)<\tau}} (1-w_{i,j}) I((i,j) \notin G_2') \right]$$

$$= \sum_{G_{2,\tau}' \subseteq G_{2,\tau}} \left[\prod_{(i,j)\in G_{2,\tau}} w_{i,j}^{I((i,j)\in G_{2,\tau}')} (1-w_{i,j})^{I((i,j)\notin G_{2,\tau}')} \right] \cdot \sum_{G_{2,\hat{\tau}}' \subseteq G_{2,\hat{\tau}}} \left[\prod_{(i,j)\in G_{2,\hat{\tau}}} (1-w_{i,j}) I((i,j) \notin G_{2,\hat{\tau}}') \right] \quad (34)$$

$$= \sum_{G_{2,\hat{\tau}}' \subseteq G_{2,\hat{\tau}}} \left[\prod_{(i,j)\in G_{2,\hat{\tau}}} (1-w_{i,j}) I((i,j) \notin G_{2,\hat{\tau}}') \right] \quad (35)$$

$$= \prod_{(i,j)\in G_{2,\hat{\tau}}} (1-w_{i,j}). \quad (36)$$

Authors' contributions

XZ proved the theorems and did the algorithm design and experiment. WW and WX contributed to the problem formulation and organized this research. All authors read and approved the final manuscript.

Competing interests

The authors declare that they have no competing interests.

Acknowledgements

This work was supported in part by the China National Science Foundation (CNSF) under Grant No. F020809.

Received: 12 December 2014 Accepted: 26 May 2015

Published online: 19 October 2015

References

1. Shah, D, Zaman, T: Rumors in a network: who's the culprit? IEEE Trans. Inf. Theory. **57**(8), 5163–5181 (2011)
2. Shah, D, Zaman, T: Rumor centrality: a universal source detector. In: Proceedings of the 12th ACM SIGMETRICS/PERFORMANCE Joint International Conference on Measurement and Modeling of Computer Systems, pp. 199–210. ACM, New York, (2012)
3. Dong, W, Zhang, W, Tan, CW: Rooting out the rumor culprit from suspects. In: 2013 IEEE International Symposium on Information Theory, pp. 2671–2675. IEEE, New York, (2013)
4. Wang, Z, Dong, W, Zhang, W, Tan, CW: Rumor source detection with multiple observations: fundamental limits and algorithms. In: Proceedings of the 2014 ACM International Conference on Measurement and Modeling of Computer Systems - SIGMETRICS '14, pp. 1–13. ACM, New York, (2014)
5. Karamchandani, N, Franceschetti, M: Rumor source detection under probabilistic sampling. In: 2013 IEEE International Symposium on Information Theory, pp. 2184–2188. IEEE, New York, (2013)
6. Luo, W, Tay, WP, Leng, M: Identifying infection sources and regions in large networks. IEEE Trans. Signal Process. **61**(11), 2850–2865 (2013)
7. Prakash, BA, Vreeken, J, Faloutsos, C: Spotting culprits in epidemics: how many and which ones? In: 2012 IEEE 12th International Conference on Data Mining, pp. 11–20. IEEE, New York, (2012)
8. Mannila, H, Terzi, E: Finding links and initiators: a graph-reconstruction problem. In: Proceedings of the 2009 SIAM International Conference on Data Mining - SDM'09, pp. 1209–1219. SIAM, Philadelphia, (2009)
9. Lappas, T, Terzi, E, Gunopulos, D, Mannila, H: Finding effectors in social networks. In: Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '10, pp. 1059–1068. ACM, New York, (2010)
10. Kempe, D, Kleinberg, J, Tardos, E: Maximizing the spread of influence through a social network. In: Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '03, pp. 137–146. ACM, New York, (2003)
11. Valiant, LG: The complexity of enumeration and reliability problems. SIAM J. Comput. **8**(3), 410–421 (1979)
12. Leskovec, J, Huttenlocher, D, Kleinberg, J: Predicting positive and negative links in online social networks. In: Proceedings of the 19th International Conference on World Wide Web - WWW '10, p. 641. ACM, New York, (2010)
13. Leskovec, J, Huttenlocher, D, Kleinberg, J: Signed networks in social media. In: Proceedings of the 28th International Conference on Human Factors in Computing Systems - CHI '10, p. 1361, New York, NY, USA, (2010)