**SCIENCE CHINA**
Information Sciences

• **Supplementary File** •

# Network topology inference from incomplete observation data

Peng DOU[1], Guojie SONG[1*] & Tong ZHAO[1]

[1]*Key Laboratory of Machine Perception(MOE), Peking University, Beijing* 100871*, China*

## Appendix A Probabilistic model of information diffusion.

Our method is based on Continuous-Time Independent Cascade Model [1] which indicates that each node $v$ can be infected by at most one neighbor $u$. The diffusion trace forms a directed transmission tree $T$, which is also a subgraph of $G$. When node $u$ gets infected, it immediately attempts to activate all of its uninfected neighbors with the probability $\gamma_{uv}$. If the neighbor $v$ is successfully activated, the information is transmitted between $u$ and $v$ after a period of time delay, denoted as $\Delta t_{uv}$. We define $P(u,v)$ as the likelihood that node $u$ transmits the information to node $v$. To simplify the problem we assume that $P(u,v)$ follows Exponential law:

$$P(u,v) \propto \alpha e^{-\alpha \Delta t_{uv}} = \alpha e^{-\alpha(t_v - t_u)}. \tag{A1}$$

There are many possible directed transmission trees that can lead to a cascade. According to NetInf, we only consider the most likely transmission tree as approximation to reduce the computation complexity. Given a complete cascade $\mathbf{t}$, the likelihood that a cascade spreads along the specific directed transmission tree $T$ is:

$$f(\mathbf{t};T) = \prod_{(u,v)\notin T,(u,v)\in G} (1-\gamma_{uv}) \prod_{(u,v)\in T,(u,v)\in G} \gamma_{uv} P(u,v). \tag{A2}$$

## Appendix B Network inference problem on incomplete cascades.

Network inference on incomplete cascades (**NIIC**) aims to infer the structure of the network underlying these incomplete cascades set, denoted by $\widetilde{C}$. With the assumption that the number of edges in the underlying network is bounded by a given number $m$, NIIC could be formalized as maximizing the likelihood function $f(\widetilde{\mathbf{t}};G)$:

$$\begin{aligned} G &= argmax_{|G|\leqslant m} f(\widetilde{C};G) \\ &= argmax_{|G|\leqslant m} \sum_{\widetilde{\mathbf{t}}\in\widetilde{C}} f(\widetilde{\mathbf{t}};G). \end{aligned} \tag{B1}$$

Given an incomplete cascade $\widetilde{\mathbf{t}}$, there is a set of possible complete cascades $D(\widetilde{\mathbf{t}})$ that can lead to $\widetilde{\mathbf{t}}$. Define $P(\mathbf{t}|D(\widetilde{\mathbf{t}}))$ as the probability that one possible complete cascades $\mathbf{t}$ happens among $D(\widetilde{\mathbf{t}})$, thus we have:

$$f(\widetilde{\mathbf{t}};G) = \sum_{\mathbf{t}\in D(\widetilde{\mathbf{t}})} P(\mathbf{t}|D(\widetilde{\mathbf{t}})) f(\mathbf{t};G). \tag{B2}$$

Based on the theoretical result of NetInf, submodularity of **NIIC** problem can be proved:

**Theorem 1.** Given the nodes set $V$ and cascades $C$, the likelihood function $f(\widetilde{\mathbf{t}};G)$ is a submodular function: $2^W \to V\times V$

*Proof.* We start by considering a complete cascade $\mathbf{t}$, graphs $G \subseteq G'$ and an edge $e=(u,v) \notin G'$. Gomez-Rodriguez et al. [2] has proved that $f(\mathbf{t};G)$ is a submodular function, that is:

$$f(\mathbf{t};G\cup e) - f(\mathbf{t};G) \geqslant f(\mathbf{t};G'\cup e) - f(\mathbf{t};G'). \tag{B3}$$

As $P(\mathbf{t}|D(\widetilde{\mathbf{t}}))$ is the same both in $G$ and $G'$, $f(\widetilde{\mathbf{t}};G)$ is *submodular* because nonnegative linear combinations of submodular functions are submodular.

* Corresponding author (email: gjsong@pku.edu.cn)

## Appendix C Generate possible cascades set $D(\widetilde{\mathbf{t}})$

Given the incomplete cascade $\widetilde{\mathbf{t}}$ in the Monte-Carlo simulation, we aim to recover the infection time of $\widetilde{V}$ based on observed infection time and produce a set of simulated complete cascades $\mathbf{t}$. We assume that the time of source node $s$ in each cascade must be observed.

During the simulation process, the activation time of an newly activated node is sampled as follows:

$$t_v = t_u + \frac{-log(1-U)}{\alpha}. \tag{C1}$$

*Proof.* According to inverse transform sampling method,

$$U = \int_0^{\Delta t_{uv}} \alpha e^{-\alpha t} dt = 1 - e^{-\alpha \Delta t_{uv}}.$$

$$e^{-\alpha \Delta t_{uv}} = 1 - U \Rightarrow \Delta t_{uv} = \frac{-\log(1-U)}{\alpha}.$$

After the simulation process, it's possible that there are still some inactive nodes after simulation, because their neighbors in $G_{i-1}$ fail to transmit the information to them. In some cases, an unobserved node $w$ in cascade $c$ can be reached by 2 or more observed nodes in $G_{i-1}$. That is, node $u_1$ and $u_2$'s infection time in $c$ have been actually observed while $w$ doesn't have time stamp in $c$, and there exist a path from $u_1$ to $w$ and a path from $u_2$ to $w$. If spanning trees $T(u_1)$ and $T(u_2)$ both contain node $w$, we just take in the earlier simulated time stamp of $w$ and keep it in corresponding spanning tree. The trace of $w$ in the other spanning tree is erased.

## Appendix D Approximate the likelihood of incomplete cascade $f(\widetilde{\mathbf{t}}; G)$

Let $\phi(\widetilde{\mathbf{t}}; G)$ be the probability density function of $D(\widetilde{\mathbf{t}})$, we can easily get the true value of $f(\widetilde{\mathbf{t}}; G)$ as follows:

$$f(\widetilde{\mathbf{t}}; G) = \int_{\widetilde{\mathbf{t}}} \phi(\widetilde{\mathbf{t}}; G) f(\widetilde{\mathbf{t}}; G) \prod_{i \in \widetilde{V}} dt_i. \tag{D1}$$

Note that under the mild regularity conditions, the Monte-Carlo approximation of $f(\widetilde{\mathbf{t}}; G)$ will converge to the true value when $M$ is sufficiently large, which means:

$$f(\widetilde{\mathbf{t}}; G) = \int_{\widetilde{\mathbf{t}}} \phi(\widetilde{\mathbf{t}}; G) f(\widetilde{\mathbf{t}}; G) \prod_{i \in \widetilde{V}} dt_i \approx \frac{1}{M} \sum_{i=1}^{M} f(\mathbf{t}_i; G). \tag{D2}$$

As discussed in NetInf, $f(\mathbf{t}; G)$ is the likelihood function as the cascade $\mathbf{t}$ spreads on the network. Many possible transmission trees can lead to $\mathbf{t}$, but it considers only the most likely transmission tree $T^*$ as approximation:

$$f(\mathbf{t}; G) \approx \max_{T \in T(G)} \log(f(\mathbf{t}; T)) = \log(f(\mathbf{t}; T^*)), \tag{D3}$$

where $T(G)$ is the set consists of all the possible directed transmission trees. $f(\mathbf{t}; T^*)$ has been defined in Eq. A2. Thus,

$$f(\widetilde{\mathbf{t}}; G) = \frac{1}{M} \sum_{i=1}^{M} log(f(\mathbf{t}_i; T^*)). \tag{D4}$$

**Finding $T^*$ based on simulation traces.** The spanning trees in simulation process can help to find the most likely transmission tree $T^*$. One simulation of one cascade leaves us with $|V \setminus \widetilde{V}|$ spanning trees: $T(u_1)$,$T(u_2)$,...,$T(u_{|V \setminus \widetilde{V}|})$. We consider firstly to find the parents of all the root nodes $u_1, u_2, ..., u_{|V \setminus \widetilde{V}|}$ except source $s$ and connecting all spanning trees into one new tree, namely **Maximum Combination Tree** $\tilde{T}$. And we prove that the maximum combination tree is the most likely transmission tree.

**Definition 1** (Maximum Combination Tree)**.**

$$\tilde{T} = \bigcup_{u \in V \setminus \widetilde{V}} T(u) \cup (par^*(u), u)$$

$$par^*(u) = argmax_{v \neq u, t_v < t_u} P(v, u),$$

where $par^*(u)$ is the node that infects $u$. The parent $par^*(u)$ of each observed node $u$ is the node that maximizes the probability $P(par(u), u)$. Note that for each node $u$ from $V \setminus \widetilde{V}$, $par^*(u)$ and $u$ do not share a same spanning tree, since $t_{par^*(u)} < t_u$ and $u$ is the root of $T(u)$. The directed edge $(par^*(u), u)$ combines two different spanning tree together. And the likelihood of $\mathbf{t}$ along $\tilde{T}$ is:

$$log(f(\mathbf{t}; \tilde{T})) = \sum_{u \in V \setminus \widetilde{V}} f(\mathbf{t}; T(u)) + \sum_{u \in V \setminus \widetilde{V}} P(par^*(u), u). \tag{D5}$$

The algorithm to find the maximum combination tree $\tilde{T}$ can be described as the following three steps, also seen in Algorithm D1:

(1) initialize $\tilde{T}$ with the combination of all simulated spanning trees $T(u_1)$,$T(u_2)$,...,$T(u_{|V \setminus \widetilde{V}|})$

(2) find the parent $par(u)$ for each observed node $u$

(3) combine $\tilde{T}$ and edge $(par(u), u)$

**Lemma 1.** For each cascade $\mathbf{t}$, the Maximum Combination Tree $\tilde{T}$ and the most likely transmission tree $T^*$ are the same.

*Proof.* Assuming $\tilde{T} \neq T^*$, there is at least one node $u$ so that

$$(x,u) \in T^*, (y,u) \in \tilde{T}, x \neq y.$$

Notice that $y = argmax_{v \neq u, t_v < t_u} P(v,u)$ and $x \neq y$, so $P(x,u) < P(y,u)$ and thus $f(\mathbf{t}, T^*) < f(\mathbf{t}, \tilde{T})$, which means $T^*$ is not the most likely transmission tree. Therefore, we have $\tilde{T} = T^*$

**Algorithm D1** Greedy-NIIC

$G_0 = null$;
**for** i=1, 2, ..., m **do**
  **for** each edge $e \notin G_{i-1}$ **do**
    **for** each $\widetilde{\mathbf{t}} \in \widetilde{C}$ **do**
      **for** j=1, 2, ..., M **do**
        **while** each node $u$ that is activated in $\widetilde{\mathbf{t}}$ **do**
          **if** $d_u = 0$ **then**
            continue; //$d_u$ is node $u$'s degree
          **end if**
          **for** each node $v$ $\{v|(u,v) \in G_{i-1} \cup e$ & $t_v = null\}$ **do**
            u tries to infect v and samples $t_v$;
          **end for**
        **end while**
        return $T(u_1)$,$T(u_2)$,...,$T(u_{|V \setminus \widetilde{V}|})$;
        $T^* = \bigcup_{u \in V \setminus \widetilde{V}} T(u)$;
        **for** each $u \in V \setminus \widetilde{V}$ **do**
          $par^*(u) \leftarrow u$;
          $P(u,u) = 0$;
          **for** each $v \in V, t_v < t_u$ **do**
            **if** $P(v,u) > P(par^*(u), u)$ **then**
              $par^*(u) \leftarrow v$;
            **end if**
          **end for**
          **if** $par^*(u) \neq u$ **then**
            $T^* = T^* \cup (par^*(u), u)$;
          **end if**
        **end for**
        return $T^*$;
      **end for**
    **end for**
    $L_e = \sum_{\widetilde{\mathbf{t}} \in \widetilde{C}} f(\widetilde{\mathbf{t}}; G_{i-1} \cup e) - f(\widetilde{\mathbf{t}}; G_{i-1})$
  **end for**
  $e_i = argmax_{e \notin G_{i-1}} L_e$;
  $G_i = G_{i-1} \cup e_i$
**end for**
return $G_m$;

## Appendix E Discussion

The Greedy algorithm starts from an empty graph $G_0$. In iteration 1, we independently consider every possible under-test edge $e$ being added into $G_1$ and compute the marginal gain. Here $f(\widetilde{\mathbf{t}}; G_{i-1} \cup e)$ reduces to $f(\widetilde{\mathbf{t}}; e)$ and $f(\widetilde{\mathbf{t}}; G_{i-1})$ is 0. The simulation process begins as follows. For each incomplete cascade $\widetilde{\mathbf{t}}$ in $\widetilde{C}$, we consider the state of nodes on edge $e$. If both ends of $e$ have got time stamps in $\widetilde{\mathbf{t}}$, there is no need to do simulation as there is only one edge that can transmit message. If there are both observed node and unobserved node in $\widetilde{\mathbf{t}}$, we perform $M$ times of simulation on $\widetilde{\mathbf{t}}$ as discussed

above. If no node in $\widetilde{\mathbf{t}}$ has time stamp, there are 2 treatments. One is to regard this cascade as invalid and ignore it. This will reduce the marginal gain. For some datasets when miss rate of incomplete cascade is high, the probability increases that both ends of an edge miss time stamps. This can lead to bad choice of edge at the initial stage of iteration and affect the following edge selection. The other one treatment is to use the average of $\Delta t$ in other cascades that the time stamps are available as approximation to compute marginal gain for the all-missing case. The edge $e_i$ with the largest marginal gain is then preserved and added into $G_1$ as fixed edge.

The algorithm proceeds until it finishes $m$ iterations and $m$ edges have already been collected in $G_m$. During this in middle iteration $i$, it will update the time stamps of all missing nodes by performing the simulation process again based on new network $G_{i-1} \cup e$ and erase the old simulated time stamps of iteration $i-1$. This is because the under-test edge $e$ brings changes to network structure and can affect the topology of most likely tree in some cases. For example in $\widetilde{\mathbf{t}}$, the start node of $e$ is $u$ and $t_u$ is observable. The end node of $e$ is $v$ that doesn't have time stamp. If $v$ has one and only parent $w$ in $G_{i-1}$ whose time is observable in $\widetilde{\mathbf{t}}$. Node $u$ and $w$ compete to activate $v$. The distribution of $t_v$ can differ a lot between iteration $i-1$ and $i$, which changes $v$'s activation chronological sequence and thus affects transmission tree structure. So that the simulated time stamps in $i-1$ iteration are hard to be reused in $i$ iteration.

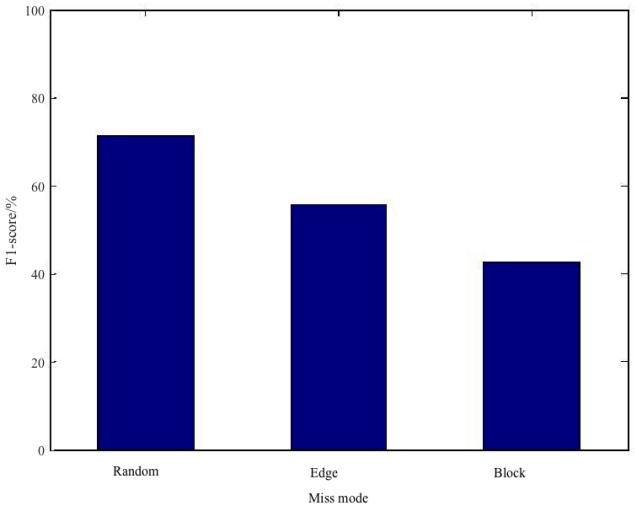


**Figure E1** Performance on miss mode.

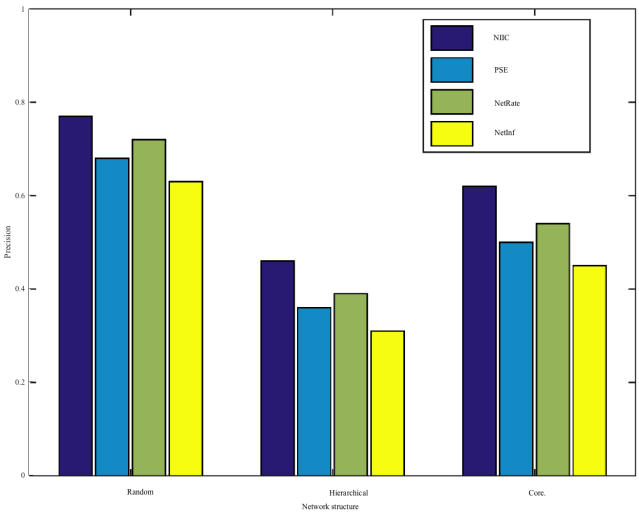


**Figure E2** Performance on network structure.

## Appendix F Experimental evaluation

In this section, we evaluate our method on both synthetic and real world incomplete data and show that our method outperforms other three widely-used baseline methods for network inference.

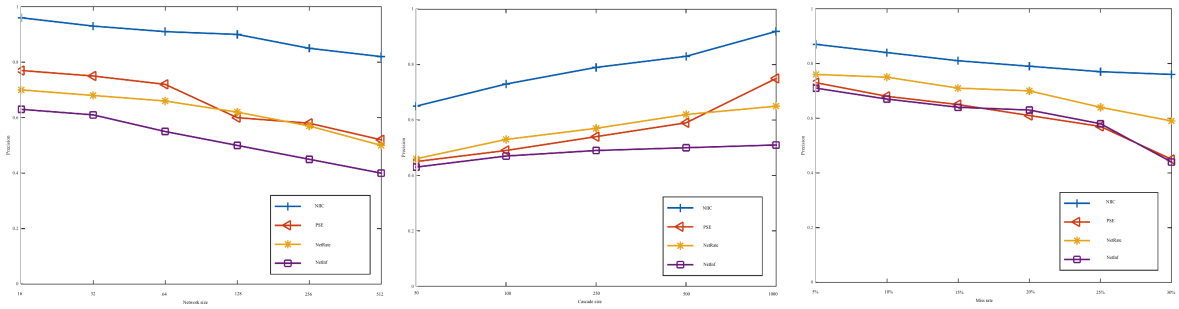

**Figure F1** Experiment results on synthetic data.

**Baseline Methods:**

- PSE [3] introduces a method to distinguish and locate the missing node and then estimates the missing time stamps. (Note that the authors haven't gave an abbreviation of their method, we call it PSE for convenience.)
- NetInf [2] introduces a tractable approximation method by exploiting a diminishing returns property of the model,.
- NetRate [4] is proposed to find the optimal network and transmission rates that maximizing the likelihood of an observed set of infection cascades and then reduces to solving a convex program.

## Appendix F.1 Experiments on synthetic data

**Experimental Setup** We consider the Kronecker Graph model to generate the diffusion networks, and we consider three different types of network structure: random(Kronecker parameter matrix[0.5,0.5;0.5,0.5]), hierarchical([0.9,0.1;0.1,0.9]), core-periphery ([0.9,0.5;0.5,0.3]). We then simulate the cascades on the generated network with the Exponential transmission model and a given transmission rate. Each time we pick the starting node uniformly at random so that at least 95% nodes are recorded in the cascades. Once a node gets infected, we record the time of the infection and mark it as infected status in case that it will be infected later.

**Performance on miss mode** Intuitively, different miss modes may lead to different performance, thus we consider the following three miss mode to evaluate our method under various conditions:

- Random Miss: Given a miss rate $q$, we randomly choose $q|V|$ nodes(infected or uninfected) from $V$ and change their infection time to $+\infty$.
- Edge Miss: Given a miss rate $q$, we randomly choose mutually disjoint $q|V|/2$ edges from $E$ and change all the infection time of nodes from these $q|V|/2$ edges to $+\infty$.
- Block Miss: Given a miss rate $q$, we firstly choose a node $u$ randomly from $V$. Then we can compute a radius $r_q$ so that $r_q = \underset{r}{argmin}\,|\{v|d_{uv} \leqslant r\}| \geqslant q|V|$. We randomly choose $q|V|$ nodes around node $u$ from $\{v|d_{uv} \leqslant r_q\}$; and change their infection time to $+\infty$.

We evaluate the solution quality by comparing the F1-score where $F_1 = 2\frac{\text{precision}*\text{recall}}{\text{precision}+\text{recall}}$. Precision is the fraction of edges in the inferred network included in the true network. Recall is the fraction of edges of the true network included in the inferred network. The Monte-Carlo simulation time is set to be 4 times of the number of network nodes. Note that the iteration time of greedy method is set to be twice of network nodes amount. Figure E1 shows the F1-score of our method on three different miss mode. We generate 500 cascades on 128 nodes with $\alpha = 1$, $\gamma = 0.2$, $q = 20\%$. When applying the random miss mode, F1-score keeps at high level of over 70%. On the hand, F1-score can also achieve 42% with block miss mode. If the unobserved nodes are connected with each other, they just form a "hidden block" instead of several nodes in the graph, of which the structure is comparatively more difficult to infer. This is why the performance based on block miss mode is worse.

**Performance on network size.** Figure F1 reveals the precision of all the methods as the networks become larger. We use the core-periphery Kronecker network and set $\alpha = 1$, $\gamma = 0.2$. There are totally 500 cascades and miss rate is 20%. We vary the number of nodes on the network from 16 to 512. The solution quality of our method is obviously better than any other one, with at least 25% increase in precision. We also observe that the accuracy declines as the network gets larger. This is due to the fact that, with the same number of cascades, smaller network makes it easier to infer.

**Performance on cascade size.** Figure F1 plots the relationship between the solution quality and the cascade size. As expected, the accuracy can be seen increasing when the number of cascades is bigger. Here, the network has 128 nodes while other variables are fixed. The precision rises from 65% to 92% for our method, while other methods can only achieve at most 77% in all cases. The precision of our method flattens out after 250 cascades, which has already been 83%. This means that, in order to achieve a good precision, a moderate number of cascades is enough.

**Performance on miss rate.** We also evaluate our method by changing the miss rate from 5% to 30%, studying how miss rate affects the results, as shown in Figure F1. In this figure,we can observe that the precision declines as miss rate increases. The gap between our method and NetRate is 6% when $q = 5\%$, while it turns to be 31% when $q = 30\%$.

**Performance on Network Structure.** From Figure E2, we can observe that the precision of our method is at least 46% in all cases and it outperforms the other three methods on different types of network structure. It also shows when

the network is based on random structure, all four methods can achieve at least 60%, better than any other two network structures, where the precision of PSE and NetInf and NetRate is lower than 40% on hierarchical model.

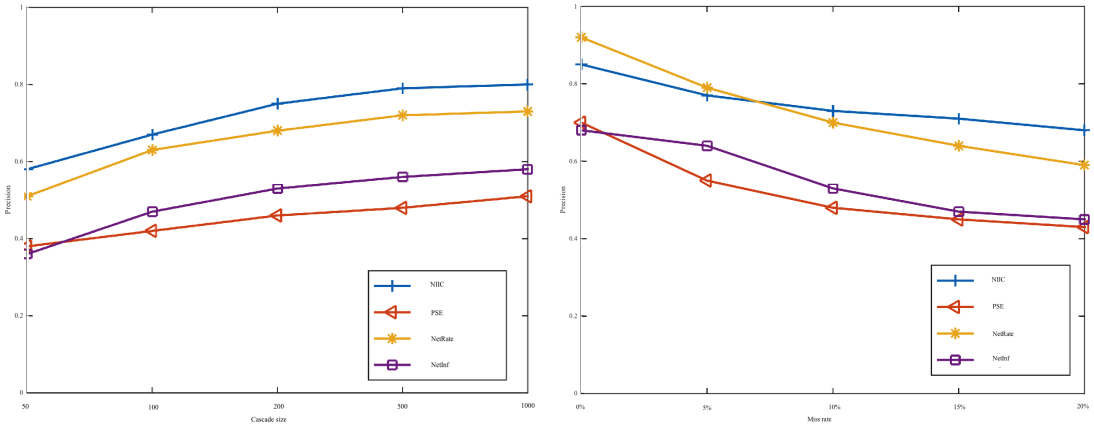


**Figure F2** Experiment results on real world data.

## Appendix F.2 Experiments on real data

**Dataset Description and Experimental Setup.** The real data we use is extracted from the quotes and phrases online that appear most frequently from August,2008 to April,2009, and the method we apply to trace information is MemeTracker. Specifically, the hyperlink between sites indicates that the piece of information is referred from one site to another. We follow the hyperlinks in order to trace the diffusion of information. For the experiment, we use the network which has 500 nodes and over 3000 edges, and the nodes we choose are the top-500 documents.

**Evaluation on real data.** Firstly, we range the miss rate from 0 to 20% and examine how the performance of our method depend on the integrity of cascades. Not surprisingly, our method provides a decent solution, as shown in Figure F2. The precision of our method as expected, are higher than any method. We can see that as the number of cascades is larger, the precision of all four methods becomes higher as expected. Our method can achieve nearly 80% accuracy while the accuracy of NetInf is no more than 60%. Figure F2 plots the precision curves and compares our results with PSE, NetRate, NetInf, as the miss rate changes. When there is no missing node, the result of NIIC and NetRate is better than that of NetInf apparently, while NIIC outperforms NetRate gradually as the miss rate increases. When the miss rate is 20%, our method can achieve over 65%, while the accuracy computed by the other three methods is below 60%.

## References

1 Kempe D, Kleinberg J, Tardos E. Maximizing the spread of influence through a social network. KDD03, 2003: 137-146
2 Rodriguez M G, Leskovec J, Krause A. Inferring networks of diffusion and influence. KDD10, 2010: 747-56
3 Dou P, Du S Z, Song G J. Inferring diffusion network on incomplete cascade data. WAIM, 2016: 325-337
4 Rodriguez M G, Balduzzi D, Scholkopf B. Uncovering the temporal dynamics of diffusion networks. ICML11, 2011: 561-568