Discover the Misinformation Broadcasting in On-Line Social Networks*

PEIGUANG LIN1, LEI CHEN2, MINGXUAN YUAN3 AND PEIYAO NIE1
1School of Computer Science and Technology
Shandong University of Finance and Economics
Jinan, 250014 P.R. China
2Department of Computer Science and Engineering
The Hong Kong University of Science and Technology
Shatin, 999077 Hong Kong
E-mail: linpg@sdufe.edu.cn

In recent years, more and more people join social networks to share information with others. At the same time, the information sharing/spreading becomes far more frequent and convenient due to the wide usage of mobile devices. As a result, the messages created are very arbitrary, which may contain a lot of misinformation. Proper actions must be taken to avoid the spreading of misinformation or rumors before it causes serious damages. Therefore, any misinformation should be discovered in time when it does not spread to a large group of people. All previous works studied either how the information is spread in the social network or how to inhibit the further pervasion of an observed misinformation. However, no works considered how to discover the broadcasting of misinformation in time. A possible solution is to set observers across the network to discover the suspects of misinformation. In this paper, we design a novel mechanism to select a set of observers in a social network with the minimum cost, where these observers guarantee any misinformation can be discovered with a high probability before it reaches a bounded number of users. Extensive experiment on real data sets verifies the effectiveness of our solution.

Keywords: social networks, random walk, propagation, misinformation, privacy

1. INTRODUCTION

The on-line social networks such as Facebook, LinkedIn, Twitter and so on are becoming more and more popular in recent years. The smart phone’s deep integration with social networks makes the information spread faster than ever. Nowadays, people can create a message by taking a picture/short video or writing a short text message with their smart phones and share it in seconds. These make both the number of users and messages very huge in current social networks. For example, the user number of Twitter in 2007-2009 is exponentially increased from less than 0.5 million to around 80 million (Detailed statistics can be found in Appendix I’s Fig. 13). The CEO of Twitter recently reported that Twitter breaks 400 million tweet-per-day barrier in 2012.

A message can be related with Anyone and Anything, around Anywhere, at Any-time. This is called 4A characteristics [16] of current social networks. Like a double blade sword, when we enjoy the benefit of rich information sharing, another serious
problem appears. The 4A characteristics make the information spread in the social network very arbitrary. Some of the information are rumors created by malicious users. For example, after Japan’s earthquake and tidal wave in March 2011, people are terrified by nuclear leak. A rumor, that eating salt can avoid the radiation, appeared and quickly spread in China’s microblog. People ran into shops/super markets to buy salt. As a result, salt was soon sold out in most cities of China. Some people even died due to salt overdose. This rumor caused a very serious consequence.

Besides malicious users, a lot of improper information may be created by users unintentionally. Some information are biased or too bloody/horrible to be known by public. For example, a person saw a crime scene and shared the bloody crime scene photos. The information may cause nightmare, public fear, or even obstruct the future investigation. We call both these two categories of messages as misinformation, which should be avoided to be shared to large populations. Another important using scenario is the negative information control by public relations firms. For example, a public relation firm needs to discover the broadcasting of the negative information about a movie star whom it served.

All previous works studied either how the information is spread in the social network or how to inhibit the further pervasion of an observed misinformation [6, 24]. However, no works considered how to discover misinformation in time. Proper actions must be taken to avoid the broadcasting of misinformation before it causes serious damages. Due to the explosion of information in current social networks, a misinformation may be initiated at any place in a network. In order to make sure any misinformation’s broadcasting be discovered in time, an efficient way is to set observers across the network to discover the suspects of misinformation. Only after a misinformation’s broadcasting is discovered, proper actions [6, 24] can be taken to suppress it.

In this paper, we would like to study how to select observers to discover the misinformation in time with the minimum cost. In other words, we would like to hire a group of observers in a large social network where each person has a cost to be hired. When an observer is hired, he/her can either use a certain filtering tool or screen all his/her information by himself/herself. Any suspicious misinformation shall be reported. We guarantee that the selected observers can find any misinformation with probability no less than \( p \) before the spreading range is too large to control (i.e. before \( b \) users got this misinformation). For example, before any misinformation is obtained by more than 1000 people, the observers can discover this misinformation with at least 80% probability. Fig. 1 shows an example of discovering the broadcasting of a misinformation initiated by \( s \). If
we hire $A$ as an observer, the broadcasting of this misinformation is observed before more than 5 people get it. If $A$ has 80% probability to get any misinformation from $s$ when that misinformation reaches less than 5 people, hiring $A$ guarantees the expected discovering probability of misinformation from $s$ before it reaches 5 peoples is at least 80%. We want to select a set of observers to guarantee the successful discovering of any misinformation initiated by any source. The successful discovering is defined by two constants $p$ and $b$. It requires that any misinformation from any source can be discovered with at least $p$ probability before the misinformation reaches $b$ users in a social network.

To solve this problem, there are two key issues:

1. How to calculate the expected discovering probability of any misinformation initiated by a user $s$ when a set of observers are given;
2. How to select the observers with the minimum cost to make sure the successful discovering.

The broadcasting of information in on-line social networks follows the Independent information Cascade Model (ICM) [6, 24]. Researchers often use ICM to study the misinformation control related issues [6, 24]. ICM is one of the most basic and well-studied diffusion models that has been used in different contexts [6, 11-15, 24]. To address the first issue mentioned above, we propose a recursive computation method to estimate the disseminate probabilities under ICM. This recursive computation method provides the upper bounds of disseminate probabilities. We also show how the accurate probabilities can be estimated by Monte Carlo Sampling. Since any user in a social network may initiate a misinformation, we need to compute the disseminate probability that any user got a misinformation from any source $s$ at any step $t$. In order to increase the computation efficiency, we further propose a top-down sampling method to reduce the number of users to be calculated.

To address the second issue, we give the formal definition of the optimal observer selection problem. With the disseminate probabilities, we can represent the expected discovering probability of any misinformation initiated by a user $s$ when a set of observers are given. We construct the optimal observer selection problem with this representation and transform it to a \{0, 1\} integer linear programing problem. Then we design an efficient approximation algorithm to solve this problem by rounding the results of its corresponding relaxed linear programing problem.

In summary, we made the following contributions:

- We study an important misinformation control problem in current on-line social networks: how to discover the misinformation broadcasting in time with a high probability. This is an important problem that has not been handled by any work before. According to our knowledge, we are the first work to solve this problem;
- We transform this misinformation control problem to a \{0, 1\} integer linear programing problem and propose an approximation solution;
- We propose proper methods to efficiently estimate the disseminate probability that any user got a misinformation from a source $s$ at any step $t$.

The rest of this paper is organized as follows: we discuss the related works in Section 2. We define the problem in Section 3 and show how we solve this misinformation
discovering problem in Section 4. Extensive experiments on real data sets are demonstrated in Section 6 followed by the conclusion in Section 7.

2. RELATED WORK

Many works studied the influence analysis of social networks [13, 23]. These works use different graphical probabilistic models and propagation algorithm to compute the influential values of users. The influential value of each user reflects how important this user is in the social network. Some other works studied how to find a set of top influential users in a social network under a budget $c$ [7, 8, 10]. Our works are different from them, we study how to discover the misinformation in time with the minimum cost in a social network.

<table>
<thead>
<tr>
<th>Table 1. Meanings of symbols used.</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p, b$</td>
</tr>
<tr>
<td>$G(V, E, W, C)$</td>
</tr>
<tr>
<td>$c_i$</td>
</tr>
<tr>
<td>$x_i$ [0, 1] variable, $x_i = 1$</td>
</tr>
<tr>
<td>$E_t(j)$</td>
</tr>
<tr>
<td>$Obv_t(j)$</td>
</tr>
<tr>
<td>$p_{t,i}(s)$</td>
</tr>
<tr>
<td>$p_{t,i}$</td>
</tr>
<tr>
<td>$a_{t,ui}$</td>
</tr>
<tr>
<td>$PRS$</td>
</tr>
<tr>
<td>$MIS$</td>
</tr>
<tr>
<td>$MNS$</td>
</tr>
<tr>
<td>$GADR$</td>
</tr>
<tr>
<td>$ADS$</td>
</tr>
<tr>
<td>$SADR$</td>
</tr>
<tr>
<td>$ASR$</td>
</tr>
</tbody>
</table>

There are many works on the spread of disease and computer virus etc. [3, 5, 9, 20] Some works studied the inhibiting of disease, virus and misinformation in a network [5, 6, 19, 24]. Meier [19] studied how to do inoculation to against the virus propagation in social networks. The owner of each node decides whether or not to protect itself [6, 24] proposed solutions, which allocate an anti-misinformation task into the social network when a misinformation is found. The new anti-misinformation task tries to propagate correct information and influence most users before the misinformation reach them. These works are based on the Independent cascade model (ICM). All the above works study how to take actions when the misinformation is observed. However, no work considered how to discover the spreading of misinformation in time. We showed in the introduction that the number of information becomes extremely large in nowadays. Thus,
discovering misinformation in time is necessary before taking actions. Our work, which studies how to discover the misinformation in time, is the pre-step of misinformation inhibiting.

Some other works studied how to decide whether a message is rumor or misinformation. Leskovec [18] used the evolution of quotes reproduced online to do this. Ratkiewicz [22] proposed a system to identify misleading political memes on Twitter using tweet features, including hashtags, links, and mentions. Qazvinian [21] studied how to decide whether a message is a rumor or not by exploring the effectiveness of three categories of features: content-based, network-based, and microblog-specific memes. Our work is different with these works, we study how to deploy the rumor detection tools on observers to make sure all the suspects of rumors are discovered in time.

3. PROBLEM DEFINITION

A social network is modeled as a weighted graph:

Definition 1: Social Network: a weighted graph represented a four-tuple $G(V, E, W, C)$, where $V$ is a set of nodes (users). $E \subseteq V \times V$ is the set of edges between nodes. $W: E \to \mathbb{R}^+$ maps each edge to a positive number between $[0,1]$. This positive number represents the probability that a message will be spread along this edge. $C: V \to \mathbb{R}^+$ maps each node to a positive number, which represents the cost to hire the corresponding user.

In the rest part of this paper, we directly use $c_i$ to represent the cost to hire user $i$. Then, the problem we need to solve in this paper is:

Problem 1:

\[
\min \sum_{i \in V} c_i x_i \\
\text{subject to:} \forall s \ E_s(s) < b \land E_{t+1}(s) \geq b \Rightarrow \text{Obv}_t(s) \geq p \quad \forall i \ x_i \in \{0, 1\}
\]

where $x_i$ is a $\{0, 1\}$ variable. $x_i = 1$ means user $i$ is selected as an observer. Otherwise, user $i$ is not selected. $E_s(s)$ means the expected number of users who get the misinformation from $s$ after $t$ step’s propagation. $\text{Obv}_t(s)$ is the expected probability that the misinformation from $s$ is observed at and before propagation step $t$. Problem 1 requires to find a set of observers such that for a misinformation initiated by any user $s$, before $b$ users get this misinformation, the expected probability that this misinformation can be observed is at least $p$. For any user $s$, if we can calculate a probability table as shown in Table 2, Problem 1 can be transformed to:

Table 2. Disseminate probability table of the misinformation from user $s$.

|   | $1$   | $2$   | ... | $i$   | ... | $|V|$ |
|---|-------|-------|-----|-------|-----|------|
| $1$ | $p_{v,1}(s)$ | $p_{v,2}(s)$ | ... | $p_{v,i}(s)$ | ... | $p_{v,|V|}(s)$ |
| ... | ...   | ...   | ... | ...   | ... | ... |
| $t$ | $p_{v,1}(s)$ | $p_{v,2}(s)$ | ... | $p_{v,i}(s)$ | ... | $p_{v,|V|}(s)$ |
| $t+1$ | $p_{v+1,1}(s)$ | $p_{v+1,2}(s)$ | ... | $p_{v+1,i}(s)$ | ... | $p_{v+1,|V|}(s)$ |
Problem 2:

\[
\min : \sum_{i=1}^{[P]} c_i \cdot x_i \\
\text{subject to:} \\
\forall s \sum_{i=1}^{[P]} p_i(s) < b \land \sum_{i=1}^{[P]} p_{i,s}(s) \geq b \Rightarrow 1 - \prod_{i=1}^{[P]} (1 - p_i(s) \cdot x_i) \geq p \\
\forall i \quad x_i \in \{0, 1\}
\]

where \( p_i(s) \), called disseminate probability, is the probability that the misinformation initiated by user \( s \) is received by user \( i \) at and before step \( t \). So the expected number of users who received the misinformation initiated by user \( s \) is \( \sum_{i=1}^{[P]} p_i(s) \). The probability that no observer gets the misinformation is \( \prod_{i=1}^{[P]} (1 - p_i(s) \cdot x_i) \). So, the probability that the observers observe the misinformation is \( 1 - \prod_{i=1}^{[P]} (1 - p_i(s) \cdot x_i) \).

4. OUR SOLUTION

In this section, we firstly discuss how Problem 2 can be solved. Then we will show how the disseminate probability table, such as Table 2, is obtained.

4.1 Optimal Observer Set Selection

Theorem 1: Problem 2 is NP-Hard.

We prove Problem 2 is NP-Hard by transforming the Weighted Set Cover Problem [12] to Problem 2. The detailed proof can be found in Appendix II.

(1) Problem Reformulation: Since \( x_i \)s are \( \{0, 1\} \) variables, we can transform Problem 2 (To be simple, we ignore the condition \( \sum_{i=1}^{[P]} p_i(s) < b \land \sum_{i=1}^{[P]} p_{i,s}(s) \geq b \) and suppose \( t \) is already given by this constraint) to:

Problem 3:

\[
\min : \sum_{i=1}^{[P]} c_i \cdot x_i \\
\text{subject to:} \\
\forall s \quad 1 - \prod_{i=1}^{[P]} (1 - p_i(s)) \geq p \\
\forall i \quad x_i \in \{0, 1\}
\]

Then we can convert the nonlinear constraints in this problem to linear constraints:

\[
1 - \prod_{i=1}^{[P]} (1 - p_i(s)) \geq p \Rightarrow \prod_{i=1}^{[P]} (1 - p_i(s)) \leq 1 - p \Rightarrow \log(\prod_{i=1}^{[P]} (1 - p_i(s))) \leq \log(1 - p) \\
\Rightarrow \sum_{i=1}^{[P]} \log((1 - p_i(s))) \leq \log(1 - p) \Rightarrow \sum_{i=1}^{[P]} \log((1 - p_i(s))) \cdot x_i \leq \log(1 - p)
\]
\[ \Rightarrow \sum_{i=1}^{p} \frac{\log((1 - p_{x_i}(s)))}{\log(1 - p)} \cdot x_i \geq 1. \]

Then Problem 3 can be transformed to the following \{0, 1\} integer linear programming (ILP) problem:

**Problem 4:**

\[
\begin{align*}
\min & \sum_{i=1}^{p} c_i \cdot x_i \\
\text{subject to:} & \\
\forall s \sum_{i=1}^{p} \frac{\log((1 - p_{x_i}(s)))}{\log(1 - p)} \cdot x_i \geq 1 & \forall i \quad x_i \in \{0, 1\}.
\end{align*}
\]

(2) **Rounding Algorithm:** \{0, 1\} integer linear programming problem is NP-Hard, which cannot be solved efficiently [1]. While if we allow \(x_i\)s to be real numbers, the linear programming (LP) problem can be solved in polynomial time. We can relax Problem 4 to a LP problem:

**Problem 5:**

\[
\begin{align*}
\min & : \sum_{i=1}^{p} c_i \cdot x_i \\
\text{subject to:} & \\
\forall s \sum_{i=1}^{p} \frac{\log((1 - p_{x_i}(s)))}{\log(1 - p)} \cdot x_i \geq 1 & \forall i \quad 0 \leq x_i \leq 1
\end{align*}
\]

We build an approximation algorithm, Algorithm 1, by rounding the result of Problem 5. We recursively solve the relaxed LP problem by a LP solver. After getting the results, we find the \(x_i\) which is the maximum among all \(x_i\)s in the current model. Then, we add user \(k\) into the selected observer set (i.e. set \(x_k = 1\)). After that, we remove variable \(x_k\) out of the LP model. If after setting \(x_k\) to be 1, a constraint in LP model is satisfied, we also remove this constraint out of the LP model. We repeat the above process until all the constraints in the LP model are removed.

**Algorithm 1:** Observer Selection Algorithm by Rounding

1. Build LP Model \(m\) follows Problem 5;
2. Set observers = \{ \};
3. while true do
4. \hspace{1cm} Solve \(m\) using a LP solvers;
5. \hspace{1cm} find \(x_k\) that \(\forall x_k \quad x_k \geq x_i\);
6. \hspace{1cm} observers = observers \(\cup\) \{\(k\)\};
7. \hspace{1cm} remove \(x_k\) out of \(m\);
8. \hspace{1cm} if \(m\) has no constrains then
9. \hspace{2cm} break;
Suppose $opt$ is the optimal solution’s cost to hire observers, and $c^*$ is the cost of Algorithm 1’s result.

**Theorem 2:** Algorithm 1’s cost $c^*$ is at most \[ (\forall i \frac{\log(1-p)}{\log((1-p_i(s)))}|V|) \] times of $opt$.

**Proof:** We use $w_i$ to represent the result (optimal function’s value) of $t$’s invocation of LP solver.

\[ w_i = w_i - c_i * x_i + c_i * x_i \]

Obviously $opt > w_i$ since the optimal solution can only take \{0, 1\} values.

\[ opt \geq w_i - c_i * x_i + c_i * x_i \]

Since $w_i \leq w_i - c_i * x_i$, (After setting $x_i = 1$, the LP solver finds a new assignment to other $x_i$’s no worse than current assignment), we can get:

\[ opt \geq w_i - c_i * x_i + c_i * x_i \geq \sum_{i=1}^{\text{observers}} c_i * x_i \geq \min(x_i) \sum_{i=1}^{\text{observers}} c_i \]

\[ = \min(x_i) \cdot c^* \]

Since $x_i$ is the maximum value of $t$’s invocation of the LP solver,

\[ \min(x_i) = \min(\forall s \max(\forall i \frac{\log(1-p)}{\log((1-p_i(s)))}|V|)). \]

For any constraint $con$, the maximum $x_i$ to make it satisfy is

\[ \max(\forall i \frac{\log(1-p)}{\log((1-p_i(s)))}|V|). \]

So the \( \min(x_i) = \min(\forall s \max(\forall i \frac{\log(1-p)}{\log((1-p_i(s)))}|V|) \).

Through our experiments (Section 6.2.3), $c^*$ is at most 30%-40% worse than $opt$ on real-life graphs.

**4.2 Disseminate Probability Table Computation**

1. Independent Cascade Model: The disseminate probabilities $p_{ij}(s)$s can be estimated under the Independent Cascade Model (ICM) [6, 24]. In the ICM [6], a process starts with an initial active node $s$, and unfolds in discrete steps. When node $u$ first becomes active in step $t$, it has a single chance to activate each currently inactive neighbor $v$; it succeeds with probability $w(u, v)$ ($w(u, v)$ is the weight on edge $e(u, v)$). If $u$ succeeds, then $v$ will become active in step $t + 1$; however, whether or not $u$ succeeds, it cannot make any further attempts in subsequent rounds. The process runs until no more activation is possible. If $v$ has incoming edges from multiple newly activated nodes, their attempts are sequenced in an arbitrary order.
(2) Recursive Estimation Method: We design a recursive formula to estimate $p_i(s)$ for any $s$. For simplicity, we directly use $p_{i,u}$ to represent $p_i(s)$ in this section. The recursive formula is:

$$p_{i+1,u} = p_{i,u} + (1 - p_{i,u}) \cdot (1 - \prod_{i \neq j}^d (1 - a_{i,v_j,u} \cdot w(u,v_j)))$$

$$a_{i+1,u,j} = (1 - p_{i,u}) \cdot (1 - \prod_{i \neq j}^d (1 - a_{i,v_j,u} \cdot w(u,v_j))) \cdot (1 - a_{j,v_i,u} \cdot w(u,v_j))$$

where $d$ is the degree of node $u$. $a_{i,v_j,u}$ is the probability that $v_j$ becomes active at step $t$ and can activate $u$ at step $t + 1$. So, the probability that $u$ receives the misinformation is $(1 - p_{i,u}) \cdot (1 - \prod_{i \neq j}^d (1 - a_{i,v_j,u} \cdot w(u,v_j)))$, where $1 - a_{i,v_j,u} \cdot w(u,v_j)$ is the probability that $v_j$ does not activate $u$ at step $t + 1$. The probability that $u$ gets the misinformation from any of its neighbors is $(1 - \prod_{i \neq j}^d (1 - a_{i,v_j,u} \cdot w(u,v_j)))$. So the probability that $u$ gets the misinformation at and before step $t + 1$ is $p_{i,u} + (1 - p_{i,u}) \cdot (1 - \prod_{i \neq j}^d (1 - a_{i,v_j,u} \cdot w(u,v_j)))$. Then, we should estimate the values of $a_{i+1,x,y}$ ($x$ and $y$ are nodes in the graph). We should make sure any $a_{i+1,x,y}$ does not contain the probability that $v_j$ gets the misinformation from its neighbor $u$. Otherwise, the computation contains the cycle that $u$ transmits the misinformation to $v_j$ at $t$ and $v_j$ transmits the misinformation to $u$ again at step $t + 1$. So, $a_{i+1,u,j} = (1 - p_{i,u}) \cdot (1 - \prod_{i \neq j}^d (1 - a_{i,v_j,u} \cdot w(u,v_j))) \cdot (1 - a_{j,v_i,u} \cdot w(u,v_j))$. In this formula, $(1 - a_{j,v_i,u} \cdot w(u,v_j))$ computes the probability that $u$ gets the misinformation at step $t + 1$ from other neighbors than $v_j$. $(1 - a_{i,v_j,u} \cdot w(u,v_j))$ is the probability that $v_j$ does not transmit the misinformation to $u$ at step $t + 1$. So, the probability that $v_j$ can get the misinformation from $u$ is $a_{i,v_j,u} = (1 - p_{i,u}) \cdot (1 - \prod_{i \neq j}^d (1 - a_{i,v_j,u} \cdot w(u,v_j))) \cdot (1 - a_{j,v_i,u} \cdot w(u,v_j))$. When $s$ initiates a misinformation, we set $p_{0,s} = 1$ and $\forall e(s,v), a_{e,s,v} = 1$. All the other $p_{0,s}$ and $a_{e,s,v}$ are set as 0. For example, Fig. 2 shows an example of misinformation spreading. The misinformation is initiated by node 0. We showed all the possible misinformation spreading cases and their...
probabilities in the figure. Table 3 shows the computation process using Formula 2. The recursive computation method accurately computes the disseminate probabilities.

Table 3. Probability computation of an acyclic graph.

<table>
<thead>
<tr>
<th>t = 0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>p₀,₁</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>a₀,₁,₂</td>
<td>1</td>
<td>a₀,₂,₁</td>
<td>0</td>
</tr>
<tr>
<td>a₀,₃,₁</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>t = 1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>p₁,₁</td>
<td>0</td>
<td>a₁,₂,₁</td>
<td>0</td>
</tr>
<tr>
<td>a₁,₃,₁</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>t = 2</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>p₂,₁</td>
<td>0</td>
<td>a₂,₂,₁</td>
<td>0.125</td>
</tr>
<tr>
<td>a₂,₃,₁</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>t = 3</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>p₃,₁</td>
<td>0</td>
<td>a₃,₂,₁</td>
<td>0</td>
</tr>
<tr>
<td>a₃,₃,₁</td>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Theorem 3: The disseminate probabilities computed based on Formula 2 are accurate when \( t \leq 3 \) and are the upper bounds of disseminate probabilities when \( t > 3 \).

Obviously, when \( t \leq 3 \), no message is cascaded in a cycle (Except the source node, since \( p_s,s = 1 \), the cycle does not influence \( s \)). So, the result of Formula 2 is accurate when \( t \leq 3 \). When \( t > 3 \), the misinformation may be cascaded in a cycle, Formula 2 may overestimate the probability that a node may get the misinformation from himself. So Formula 2 computes the upper bounds of disseminate probabilities when \( t > 3 \).

(3) Monte Carlo Sampling Method: To accurately calculate the disseminate probability table, the most common method is to use Monte Carlo Sampling method. We simulate the process of ICM by randomly activating neighbors. Each time when a node \( u \) becomes active, for any edge \((u, v)\), we randomly active \( v \) (in case \( v \) has not gotten the misinformation) with probability \( w(u, v) \).

5. REDUCE THE COMPUTATION COMPLEXITY

To solve the optimal observer set selection problem, we need to compute a disseminate probability table for each node \( s \) in the graph. The computation cost may become heavy when the graph is large. This also makes the optimal problem to be solved (i.e. Problem 3) more complex, which has \(|V|\) constraints. In order to reduce the computation cost, we can use one node to represent several nodes. We use a top-down sampling method to select a subset of nodes. Each selected node represents a group of nodes in the original graph. Thus, we only compute the disseminate probability tables of these selected nodes.

We design a top-down sampling method. We show the working process of our method in Fig. 3. We recursively divide the graph into subgraphs until the size of each subgraph is smaller than a threshold \( \text{maxSize} \). When a subgraph \( g \)'s size is larger than \( \text{maxSize} \), we randomly divide \( g \) into two subgraphs \( g_1 \) and \( g_2 \). Then we swap the nodes between \( g_1 \) and \( g_2 \) until the number of edges between these two subgraphs is minimized.
6. EXPERIMENT

6.1 Experiment Set-up

6.1.1 Data Sets: We test our solution on two real data sets

- **Facebook Data (FB)**
  Facebook (facebook.com) is the most popular used social network nowadays. We extract a subgraph of the Facebook, which contains 1422 nodes and 29995 edges. (The edge represents the friendship relations between users. The weights on the edges represent the probability of posting on each other’s wall. They are computed from the frequencies of history postings.) The average degree is 42.2. The reason we choose a subgraph is because that this subgraph represents the basic connection characteristics of Facebook. Our purpose is to study how information covers a subset of users in a social network. The Monte Carlo Sampling Method can be easily operated on the subgraph to get the accurate disseminate probability table as ground truth. The experiments on subgraph can represent the effectiveness on full graph.

  After we select a subset of representing nodes, Problem 3 becomes:

  **Problem 6:**

  \[
  \begin{align*}
  \min : & \quad \sum_{j=1}^{\vert V \vert} c_j \cdot x_j \\
  \text{subject to:} & \quad \forall s \in \text{selected} \sum_{i=1}^{\vert V \vert} \frac{\log((1 - p_{ij}(s)))}{\log(1 - p)} \cdot x_i \geq 1 \quad \forall i, x_i \in \{0, 1\} 
  \end{align*}
  \]

- **Arnet Data (AN)**
  ArnetMiner (http://www.arnetminer.net/) is an academic researcher social network collected by Tsinghua University. The Arnet contains information extracted from crawl-
ed web pages of computer scientists. The extracted information forms the attribute values of individual researchers, such as contact information, affiliation and research interests. In addition, the data also contains the co-authorship graph between these people. The weights on the edges are computed from the cooperation strength between researchers. We extract a subgraph from ArnetMiner, which contains 6000 nodes and 37848 edges. The average degree is 12.6.

### 6.1.2 Comparison methods

In order to demonstrate the effectiveness of our solution, we compare our results with the following three basic methods. Suppose our algorithm finds an observer set with a hiring cost $c^*$, we generate the other three observer sets by:

- **Pure Random Selection (PRS):**
  We randomly select observers until the hiring cost exceeds $c^*$. Expectedly, the random selection has a chance to distribute the observers evenly in the social network. We also call this method PRS method.

- **Maximum Influence Node Selection (MIS):**
  We use the pageranking method [25, 26] to compute the influencing values of each node. After that, we select the observers, which have the maximum sum of influent values with the cost at most $c^*$. We also call this method MIS method.

- **Maximum Node Selection (MNS):**
  We select the observer set, which contains the maximum number of observers with the cost at most $c^*$. We also call this method MNS method.

We compare the discovering effects of misinformation broadcasting of these three methods and our four methods:

- **Full-Recursive:** Full Computation on Recursive Disseminate Probability Estimation Method
- **Full-Monte Carlo:** Full Computation on Monte Carlo Disseminate Probability Estimation Method
- **Sampling-Recursive:** Sampling Computation (Algorithm 2) on Recursive Disseminate Probability Estimation Method
- **Sampling-MonteCarlo:** Sampling Computation (Algorithm 2) on Monte Carlo Disseminate Probability Estimation Method

### 6.1.3 Criteria

We let each user in a social network initiates 100 misinformation and broadcast randomly. Then there are totally $100|V|$ misinformation. We test the discovering effects by four criteria:

- **Global Average Discovering Ratio (GADR)**
  For a misinformation $m$, suppose when $m$ is observed, the set of users who get this
message is \( R(m) \). Then if \(|R(m)| < b\), \( m \) is successfully discovered. We represent it as \( o(m) = 1 \). Otherwise, \( o(m) = 0 \). The global average discovering ratio (GADR) can be computed as:

\[
GADR = \frac{\sum_{m} o(m)}{100 |V|} \times 100\%.
\]

GADR represents that how many misinformation are successfully discovered globally. Larger GADR is preferred.

- **Average Discovering Size (ADS)**
  
  We compute the average discovering size (ADS) as:

\[
ADS = \frac{\sum_{m} |R(m)|}{100 |V|} \times 100\%.
\]

Note if in an extreme case, when a misinformation \( m \) finished its broadcasting, it is still not observed, we use the size of users who finally get i as \(|R(m)|\).

ADS compute the average size of influenced users by a misinformation. Smaller ADS is preferred.

- **Single Average Discovering Ratio (SADR)**
  
  For any source user \( s \), we compute the discovering ratio of the misinformation initiated by \( s \) as:

\[
DR(s) = \frac{\sum_{m \text{ from } s} o(m)}{100}.
\]

The single average successful discovering ratio (SADR) can be calculated as:

\[
SADR = \frac{\sum_{s} DR(s)}{|V|} \times 100\%.
\]

SADR computes the average discovering probability estimated by each creating user. Larger SADR is preferred.

- **Average Successfull Ratio (ASR)**
  
  For any source user \( s \), if \( ASR(s) \geq p \), the misinformation initiated by \( s \) is successfully discovered. We represent it as \( d(s) = 1 \). Otherwise, \( d(s) = 0 \). Then the average successfully ratio is computed as:

\[
ASR = \frac{\sum_{s} d(s)}{|V|} \times 100\%.
\]

ASR computes the average ratio of successfully discovered misinformation estimated by each creating user. Larger ASR is preferred.
Besides the above four criteria, we also tested the effectiveness of Algorithm 1.

6.2.1 Discovering effects

We tested the misinformation discovering effectiveness based on different observer set selection methods. We use the Linear Programming Solver \textit{lp solve} [2]. We set $b = 0.1|V|$ and change $p$ from 0.4 to 0.9. Figs. 5 and 9 compares the Full-Recursive method with the three basic methods on data set FB and AN respectively. Figs. 4 (a) and 8 (a) shows the result on \textit{GADR}. From the figures, we can see Full-Recursive method guarantees very high global average discovering ratio for both two data sets. For FB, the \textit{GADR} of the Full-Recursive method is near to 100%. For AN, the \textit{GADR} of the Full-Recursive method is larger than 95% in most cases. The value of \textit{GADR} is bigger than the corresponding threshold $p$. This reflects the safety of Full-Recursive method (The recursive method computes the upper bounds of the disseminate probabilities). Another result we can observe is that with the same budget, our selection method, which has larger \textit{GADR}, overcomes the three basic methods.

![Graphs](attachment:graphs.png)

Fig. 4. FB: Full-Recursive (Full Computation on Recursive Estimation Method).

Similar to \textit{GADR}, the \textit{SADR} (Fig. 4 (c) for FB and Fig. 8 (c) for AN) and \textit{ASR} (Fig. 4 (d) for FB and Fig. 8 (d) for AN) are also near to 100%. These also confirm the safety of Full-Recursive method. The \textit{GADR}, \textit{SADR} and \textit{ASR} of Full-Recursive method is much larger than the three basic selection methods. The \textit{ADS} (Fig. 4 (b) for FB and Fig. 8 (b) for AN) is much smaller than the three basic methods. That means with the same budget, our method achieves much better discovering effectiveness.
DISCOVER THE MISINFORMATION BROADCASTING IN ON-LINE SOCIAL NETWORKS

Figs. 6 and 9 compare the Full-Monte Carlo method with the three basic methods on data set FB and AN respectively. We can observe the similar results to Full-Recursive method, especially on FB. These results reflect that in real social networks, we can use the recursive method to quickly estimate the disseminate probabilities instead of the Monte Carlo method, which has much higher cost. In Section 6.2.2, we will show the estimation error of the recursive method compared to the Monte Carlo method in detail. From the results, we can also find that the Full-Monte Carlo method is safe and our selection method achieves much better discovering effectiveness than the three basic methods.

![Graphs showing comparison](image)

Fig. 5. FB: Full-Monte Carlo (Full Computation on Monte Carlo Method).

We tested the effectiveness of Sampling-Recursive method and Sampling-Monte-Carlo method by selecting 256 nodes of FB and AN respectively using Algorithm 2. That means we use 18% nodes of FB and 4% nodes of AN to do the observer selection. The sampling methods take only less than 20% running time of the full computations on FB and less than 1% on AN to calculate the observer set. Figs. 7 and 11 compares the Sampling-Recursive method with the three basic methods. While, Figs. 8 and 12 compares the Sampling-Monte Carlo method. From the results, we can observe that using the sampling decreases the discovering effectiveness a little bit. However, the selection results are still quite good. The GADR, SADR and ASR are bigger than 80% in most cases. The results of Sampling-Recursive method are also quite similar to the Sampling-Monte Carlo method. From the figures, we can also see that our methods defeat the three basic methods.

To summarize, we can conclude that the Sampling-Recursive method is the best method when we have limited time to select observers. While, the Full-Recursive method is the best method when we have adequate time.
6.2.2 Accuracy of the recursive disseminate probability estimation method

In this section, we test how accurate the recursive method can estimate the disseminate probabilities compared to the Monte Carlo method. Suppose $p^R_{i,s}(s)$ is the disseminate probability estimated by the recursive method and $p^M_{i,s}(s)$ is the disseminate proba-
bility estimated by the Monte Carlo method, we compute the error as

$$\text{error}_{s}(s) = \frac{p_{r}^{s}(s) - p_{m}^{s}(s)}{p_{r}^{s}(s) + p_{m}^{s}(s)} \times 100\%.$$ 

Fig. 8. AN: Full-Recursive (Full Computation on Recursive Estimation Method).

Fig. 9. AN: Full-Monte Carlo (Full Computation on Monte Carlo Estimation Method).
We compute the average error for all disseminate probabilities and summarize the error statistics in Table 4. From the result, we can see for FB, the average error is 4.75%, which is very small. 86.85% disseminate probabilities are estimated within the error 5% and more than 90% disseminate probabilities are estimated within the error 10%-20%.
The recursive method works quite well. For AN, the average error is 30.2% and more than 50% disseminate probabilities are estimated within the error 15%. The result is quite good, too. These explain why the experiment results of recursive method and Monte Carlo method are similar in Section 6.2.1.

The reason why the recursive method works better on FB than on AN is that FB is denser than AN. The average degree of FB is 42.2, which is much larger than AN’s 12.6. The recursive method can estimate the disseminate probabilities when \( t \leq 3 \). Since FB is denser, more nodes in FB need to calculate the disseminate probabilities with \( t \leq 3 \). So, the recursive method works better on FB than AN.

![Fig. 12. The Effectiveness of Algorithm 1.](image)

### Table 4. Accuracy of the recursive disseminate probability estimation method.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Average</th>
<th>&lt; 5%</th>
<th>&lt; 10%</th>
<th>&lt; 15%</th>
<th>&lt; 20%</th>
</tr>
</thead>
<tbody>
<tr>
<td>FB</td>
<td>4.75%</td>
<td>86.85%</td>
<td>93.53%</td>
<td>96.69%</td>
<td>96.84%</td>
</tr>
<tr>
<td>AN</td>
<td>30.2%</td>
<td>28.47%</td>
<td>41.52%</td>
<td>50.55%</td>
<td>56.47%</td>
</tr>
</tbody>
</table>

#### 6.2.3 Effectiveness of Algorithm 1

In order to test how good Algorithm 1 works, we compare the hiring cost computed by Algorithm 1 with the output of the optimal \{0, 1\} integer programming (i.e., we directly use the integer programming solver \( lp\_solve \) to solve Problem 4).

Suppose the cost computed by Algorithm 1 is \( c^* \) and the optimal cost is \( opt \), we compute the value \( \frac{c^* - opt}{opt} \times 100\% \). Since Problem 4 is NP-Hard, we do this experiment on a group of subgraphs with 50-80 nodes on each data set (when the subgraph contains 100 nodes, the integer programming solver takes more than 48 hours without reach the optimal solution for some \( ps \)). Fig. 12 (a) shows the testing results on the subgraphs sampled from FB. Algorithm 1 works at most 40% worse than the optimal solution. In most cases, Algorithm 1 can make \( c^* \) within 10% bigger than the optimal solution. Fig. 12 (b) shows the testing results on the subgraphs sampled from AN. Algorithm 1 works at most 30% worse than the optimal solution. In most cases, Algorithm 1 can make \( c^* \) within 15% bigger than the optimal solution. From the results, we can see Algorithm 1 works quite well.
7. CONCLUSION

With the more and more popularity of social networks and exponential increasing of information sharing, misinformation in social networks may cause serious negative results. Thus, misinformation control becomes an important problem. Proper actions must be taken to avoid the spreading of misinformation or rumor before it is spread to a large group of people. In this paper, we design a novel mechanism to select a set of observers in a large social network with the minimum cost. These observers guarantee any misinformation can be observed with a high probability when this misinformation does not reach a bounded number of users. According to our knowledge, this is the first work to solve this problem.

REFERENCES

2. Ip solve, a Mixed Integer Linear Programming (MILP) solver, website.


**APPENDIX I: TWITTER USER NUMBER STATISTICS**

Fig. 13 shows the statistics of the user numbers in Twitter from 2007 to 2009.
APPENDIX II: NP-HARD PROOF

Theorem 4: Problem 2 is NP-Hard.

Proof: We prove Problem 2 is NP-Hard by transforming the Weighted Set Cover Problem [12] to Problem 2. Weighted Set Cover problem is: given a universe of \( n \) members \( U = e_1, ..., e_n \), \( S = \{s_1, ..., s_l\} \), \( s_i \subseteq U \) and \( \bigcup_{i=1}^{l} s_i = U \), \( c: S \rightarrow \mathbb{R}^+ \) maps each \( s_i \) to a real number cost \( c_i \). We want to find a cover \( C \subseteq S \) (i.e. \( \bigcup_{j \in C} s_j = U \)) where \( C \)'s weight \( \sum_{j \in C} c_j \) is minimized. If we use \( x_i \) to represent whether \( s_i \) is involved in \( C \) (i.e. \( x_i = 1 \iff s_i \in C \) and \( x_i = 1 \iff s_i \not\in C \)), the Weighted Set Cover Problem can be represented as the following \{0,1\} integer programming problem:

\[
\begin{align*}
\text{min} & : \sum_{i=1}^{n} c_i \cdot x_i \\
\text{subject to:} & \quad \forall e_i \sum_{\forall j: e_i \in s_j} x_j \geq 1 \quad \forall i \in \{0,1\}.
\end{align*}
\]

We can do the following transformation for each constraint in Formula 3:

\[
\begin{align*}
\sum_{\forall j: e_i \in s_j} x_j \geq 1 & \Rightarrow \sum_{\forall j: e_i \in s_j} x_j \geq 1 \\
& \Rightarrow \sum_{\forall j: e_i \in s_j} \log(1-p) \cdot x_j \leq \log(1-p) \quad (0 < p < 1) \\
& \Rightarrow \sum_{\forall j: e_i \in s_j} \log((1-p)^{x_j}) \leq \log(1-p) \\
& \Rightarrow \log(\prod_{\forall j: e_i \in s_j} (1-p)^{x_j}) \leq \log(1-p) \\
& \Rightarrow \prod_{\forall j: e_i \in s_j} (1-p)^{x_j} \leq 1-p \\
& \Rightarrow 1- \prod_{\forall j: e_i \in s_j} (1-p)^{x_j} \geq p.
\end{align*}
\]

Since \( x_j \) can only be 0 or 1, \( 1 - \prod_{\forall j: e_i \in s_j} (1-p)^{x_j} \geq p \iff 1 - \prod_{\forall j: e_i \in s_j} (1-p)^{x_j} \geq p. \) The Weighted Set Cover problem is successfully transformed to Problem 2. So Problem 2 is NP-Hard.

Peiguang Lin is currently a Researcher of Computer Science and Technology, Shandong University of Financial and Economics. He got his doctor degree from School of Computer Science and Technology, Beijing institute of Technology. His research interests include social network and data mining.
Lei Chen is currently a Researcher of Department of Computer Science and Engineering, Hong Kong University of Science and Technology. His research interests include crowdsourcing-based data processing, uncertain and probabilistic databases, web data management, multimedia and time series databases and privacy.

Mingxuan Yuan is currently a Researcher of Huawei Noah’s Ark lab, Hong Kong. Before that, he served as a PostDoc fellow in the Department of Computer Science and Engineering of the Hong Kong University of Science and Technology. His research interests include big telecom (spatiotemporal) data storage/management, telecom data mining and data privacy.

Peiyao Nie is currently a Researcher of Computer Science and Technology, Shandong University of Financial and Economics. His research interests include database and data mining.