Source Locating in Social Network

Linguo Li, Chaofei Wang, Bingkun Zhao

May 31, 2018

Abstract

The application of networks in daily life is very extensive, in which positioning the source of propagation is a very important technique. For example, find the source of the disease in infectious diseases and finding the source of rumors in the network. This article describes some methods of source locating in social network, including A Sample Path Based Approach and Gradient Maximum Likelihood Algorithm, which are respectively studied by Chaofei Wang and Linguo Li. Moreover, we also do some research on multiple sources locating problem, which mainly studied by Bingkun Zhao.

1 Introduction

We live in network society. We interact with many networks every second to collect, process, and transmit large amounts of information. The increase in the interconnectedness of the world makes it easier for us to receive false information, and it is also easier to be blinded by false information. For example, rumors on Weibo often affect the entire community of netizens. Obviously, one of the major challenges we face is to develop effective methods to detect the source, so that we can effectively suppress the spread of rumors.

Many predecessors have studied in this area. For example, in a state where the network is a regular tree, an irregular geometric tree, or the like, the rumor centrality of a node is introduced, but it is necessary to know all the connections between the nodes and the states of all the nodes. It was also suggested that only some of the node states called as observers need to be known, but they need to be monitored. However, these methods are very complicated and computationally expensive, so they are difficult to apply. This paper proposes some methods reducing the number of observers, which greatly reduces the complexity. In addition, most of the current studies on source tracing are based on single-source networks, so we studied the method of multiple source locating problem.

2 Sample Path Based Approach

2.1 Problem introduction

In real world, we will always come across the diffusion problem in networks, such as outbreak of epidemics, rumors spreading online/social network, the spreading of virus over network, the blackout of the grid, etc.

We need to use a fast method to detect the source to avoid more loss or influence. So based on this situation, there are many researchers develop various methods to help solve the problem.

Methods are based on the topology and the situation of the network. Or say such problem can be abstract into a nodes-lines problem. Nodes represent the users in social network or computers in infected networks, lines are the connections between them. The diffusion spread from node to node via lines. Given a snapshot of the diffusion process at time t, our aim is to find which node is the source of the diffusion. We called this problem information source detection problem.

2.2 Models

SIR: Susceptible-Infected-Recovered model, a standard model of epidemics. The network is assumed to be an undirected graph and each node in the network has three possible states: susceptible (S), infected(I), and recovered (R). Nodes in state S can be infected and change to state I, and nodes in state I can recover and change to state R. Recovered nodes cannot be infected again. We assume that initially all nodes are in the susceptible state except one infected node (called the information source). The information source then infects its neighbors, and the information starts to spread in the network. Now given a snapshot of the network, in which we can identify infected nodes and healthy (susceptible and recovered) nodes (we assume susceptible nodes and recovered nodes are indistinguishable), the question is which node is the information source.

Also, there are other models like SIR, SIS, SIX, etc. We only discuss the diffusion process and our technique under the SIR model.

2.3 Works

2.3.1 Previous works:

MLE: traverse all the possible diffusion path and select the one which is most likely to lead to the final result (the snapshot we got).The solving process formulate as:

$$v^+ \in \underset{v \in \upsilon}{argmax} \sum_{X[0:t]:F(X(t))=Y} Pr(X[0,t]|v^* = v) \tag{1}$$

With the complexity of $\Omega(t^N)$, where N is the network size and t is the time the snapshot is obtained. (It is difficult for us to solve the problem).

2.3.2 Ours:

Sample path based approach(one of the Jordan centers): Instead of using the MLE, we propose to identify the sample path $X^*[0,t^*]$ that most likely leads to Y, ie.,

$$X^*[0,t^*] = \underset{t,X[0,t] \in \chi(t)}{argmax} Pr(X[0,t]), \tag{2}$$

where $\chi(t) = \{X[0,t]|F(X(t)) = Y\}$. The source node associated with $X^*[0,t^*]$is then viewed as the information source.

On tree networks: It is diffucult to obtain the results on general graphs, so we focus on tree networks and derive structure properties of the optimal sample paths.

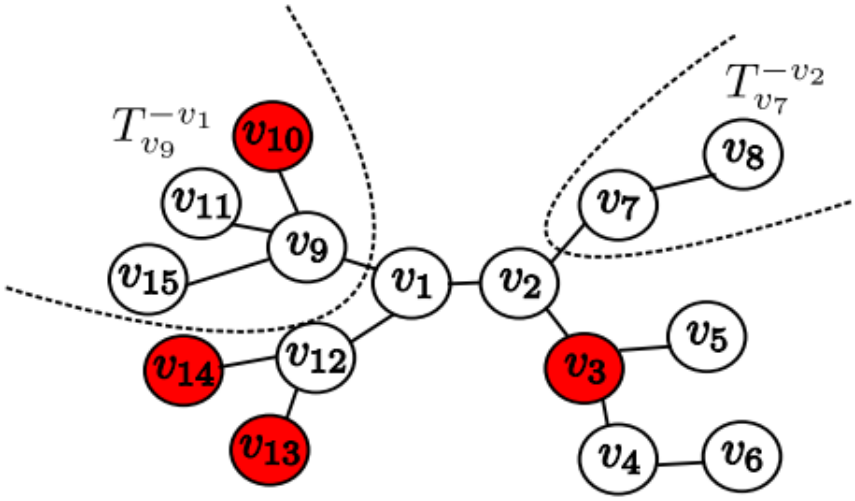

Figure 1: An Example Illustrating the Infection Eccentricity

First, we introduce the definition of eccentricity in graph theory. The eccentricity $e(v)$ of a vertex v is the maximum distance between v and any other vertex in the graph. The Jordan centers of a graph are the nodes which have the minimum eccentricity. For example, in Figure 2, the eccentricity of node v_1 is 4 and the Jordan center is v_2; whose eccentricity is 3.

Following a similar terminology, we define the infection eccentricity $\widetilde{e}(v)$ given Y as the maximum distance between v and any infected nodes in the graph. Define the Jordan infection centers of a graph to be the nodes with the minimum infection eccentricity given Y: In Figure 2, nodes v_3, v_{10}, v_{13} and v_{14} are observed to be infected. The infection eccentricities of v_1, v_2, v_3, v_4 are 2, 3, 4, 5, respectively, and the Jordan infection center is v_1.

We will show that the source associated with the optimal sample path is a node with the minimum infection eccentricity.

Reverse infection algorithm: Since in tree networks with infinitely many levels, the estimator based on the sample path approach is a Jordan infection center, we view the Jordan infection centers as possible candidates of the information source. We next present a simple algorithm to find the information source in general networks. The algorithm is to first identify the Jordan infection centers, and then break ties based on the sum of distances to infected nodes.

The key idea of the algorithm is to let every infected node broadcast a message containing its identity (ID) to its neighbors. Each node, after receiving messages from its neighbors, checks whether the ID in the message has been received. If not, the node records the ID (say v), the time at which the message is received (say t_v), and then broadcasts the ID to its neighbors. When a node receives the IDs of all infected nodes, it claims itself as the information source and the algorithm terminates. If there are multiple nodes receiving all IDs at the same time, the tie is broken by selecting the node with the smallest $\sum t_v$.

The tie-breaking rule we proposed is to choose the node with the maximum infection closeness. The closeness measures the efficiency of a node to spread informa-

tion to all other nodes. The closeness of a node is the inverse of the sum of distances from the node to any other nodes. In our model, we define the $infectioncloseness$ as the inverse of the sum of distances from a node to all infected nodes, which reflects the efficiency to spread information to infected nodes. We select a Jordan infection center with the largest infection closeness, breaking ties at random

Algorithm 1 Reverse Infection Algorithm

for $i \in \mathcal{L}$ **do**
  i sends its ID ω_i to its neighbors.
end for
while $t \geq 1$ and STOP $==$ 0 **do**
  for $u \in \upsilon$ **do**
    if u receives ω_i for the first time **then**
      Set $t_{ui} = t$ and then broadcast the message ω_i to its neighbors.
      If there exists a node who received $|\mathcal{L}|$ distinct messages, then set $STOP == 1$.
    end if
  end for
end while
return $u^+ = argmin_{u \in S} \sum_{i \in \mathcal{L}} t_{ui}$, where S is the set of nodes who receive$|\mathcal{L}|$distinct messages when the algorithm terminnates. Ties are broken at random.

It is easy to verify that the set S is the set of the Jordan infection centers. The running time of the algorithm is equal to the minimum infection eccentricity and the number of messages each node receives/sends during each time slot is bounded by its degree.

2.4 Simulation and results analysis

In this section, I will give some simulate result of the methods on different networks. Then, anlyze the results.

2.4.1 Tree Networks

In this section, we evaluate the performance of the reverse infection algorithm on tree networks. We compare the reverse infection algorithm with the closeness centrality heuristic, which selects the node with the maximum infection closeness as the information source.

1. *Small-size tree networks:* We first studied the performance on small-size trees. The infection probability q was chosen uniformly from (0,1) and the recovery probability p was chosen uniformly from (0,q). The infection process propagates $t$time slots where t was uniformly chosen from [3,5]. To keep the size of infection topology small, we restricted the total number of infected and recovered nodes to be no more than 100. For small-size trees, we first calculated the MLE using dynamic programming for fixed t and then searching over $t \in [0, t_{max}]$ for a large value of t_{max} to find the optimal estimator. The detection rate is defined to be the fraction of experiments in which the estimator coincides with the actual source. We varied g from 2 to 10 and the results are shown in Figure 2. We can see that the detection rate of the reverse infection algorithm is almost the same as that of the MLE, and is higher than that

of the closeness centrality heuristic by approximately 20% when the degree is small and by 10% when the degree is large.

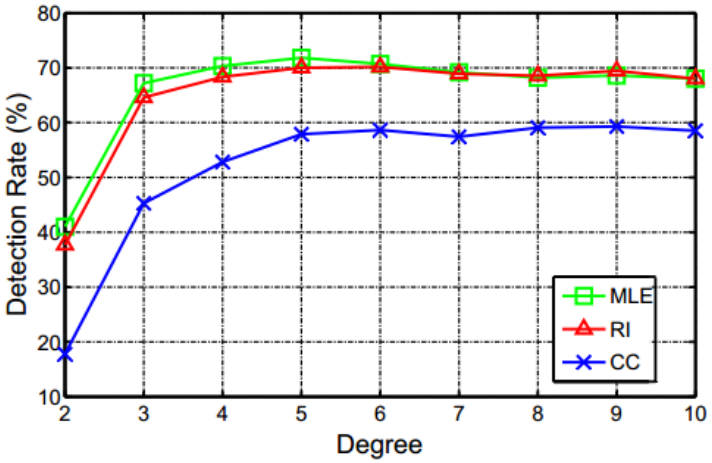

Figure 2: The Detection Rates of the Maximum Likelihood Estimator (MLE), Reverse Infection (RI) and Closeness Centrality (CC) on Regular Trees

2. *General g-regular tree networks:*We further conducted our simulations on large-size g-regular trees. The infection probability q was chosen uniformly from (0, 1) and the recovery probability p was chosen uniformly from $(0, q)$. The infection process propagates t time slots where t was uniformly chosen from [3,20]. We selected the networks in which the total number of infected and recovered nodes is no more than 500. We varied g from 2 to 10. Figure 3 shows the detection rate as a function of g. We can see the detection rates of both the reverse infection and closeness centrality algorithms increase as the degree increases and is higher than 60% when $g > 6$. However, he detection rate of the reverse infection algorithm is higher than that of the closeness centrality algorithm, and the average difference is 8.86%.

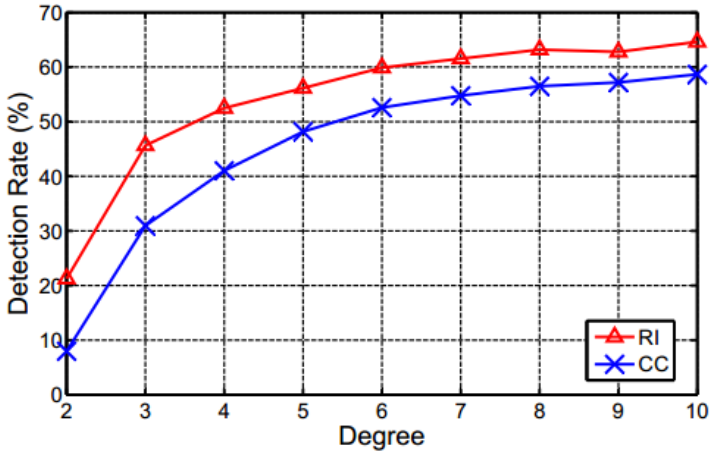

Figure 3: The Detection Rates of the Reverse Infection (RI) and Closeness Centrality (CC) Algorithms on Regular Trees

2.4.2 Real World Networks

We next conducted experiments on three real world networks the Internet Autonomous Systems network $(IAS)^3$, the Wikipedia who-votes-on-whom network

($Wikipeida$)[4], and the power grid network (PG)[4]. We compare the reverse infection algorithm with random guessing, which randomly selects a node and declares it as the information source. In these networks, the infection probability q was chosen uniformly from (0, 0.05) and the recovery probability p was chosen uniformly from (0,q). Here we chose small infection probabilities since the network was of finite size so the infection process should be controlled to make sure that not all nodes were infected when the network was observed. The duration t was an integer uniformly chosen from [3,200]. We selected the networks in which the total number of infected and recovered nodes was in the range of [50, 500].

1. *The Internet autonomous systems network:*Figure 4 shows the results on the the Internet autonomous systems network. An Internet autonomous system is a collection of connected routers who use a common routing policy. The Internet autonomous system network is obtained based on the recorded communication between the Internet autonomous systems inferred from Oregon route-views on March, 31st, 2001. The network consists of 10,670 nodes and 22,002 edges. According to Figure 3, more than 80% of the estimators identified by the reverse infection algorithm are no more than two hops away from the actual sources, comparing to 10% under the random guessing.

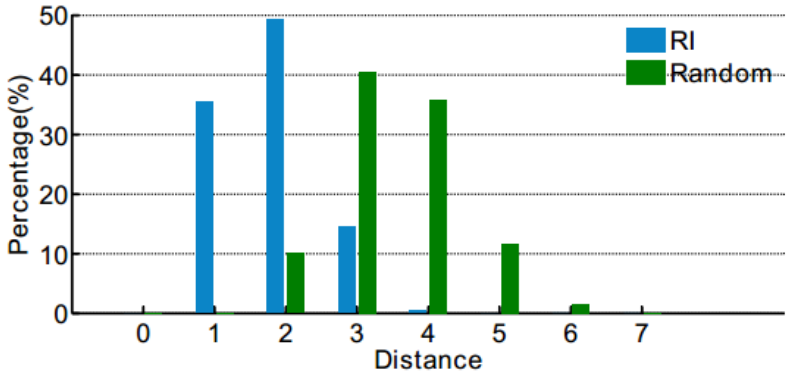

Figure 4: The Performance of the Reverse Infection (RI) on the Internet Autonomous Systems Network

2. *The Wikipedia who-votes-on-whom network:*Figure 5 shows results on the Wikipedia who-votes-on-whom network, in which two nodes are connected if one user voted on the other in the administrator promotion elections. The network has 100,736 links and 7,066 nodes. We have similar observations as for the Internet autonomous systems network, the majority of the estimators produced by the reverse infection algorithm are no more than two hops away from the actual sources and only less than 20% of the estimators of random guessing are within two hops from the actual sources.

3. *The power grid network:*Figure 6 shows the results on the power grid, which has 4,941 nodes and 6,594 edges. As we can see, the reverse infection algorithm performs better than the random guessing. The peak of the reverse infection algorithm appears at the third hop versus the seventeenth hop under random guessing.

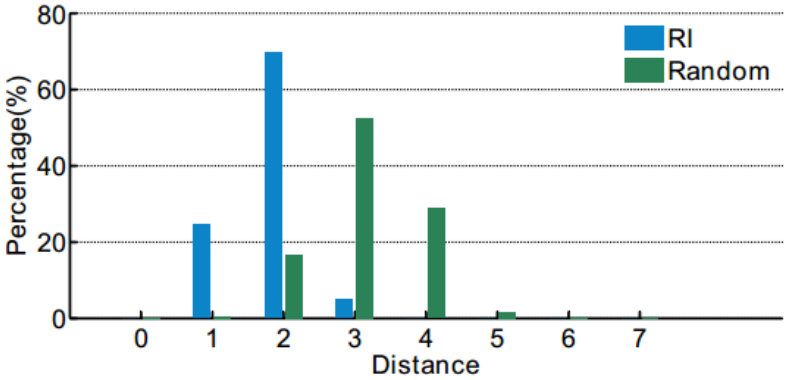

Figure 5: The Performance of the Reverse Infection (RI) on the Wikipedia Who-Votes-on-Whom Network

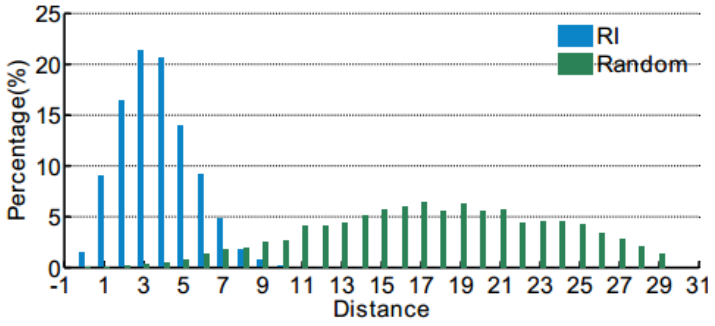

Figure 6: The Performance of the Reverse Infection (RI) on the Power Grid Network

2.5 Appendix:

If you want to kown more details about the technique and the model, you can read this part.

2.5.1 SIR model details:

Consider an undirected graph $G = \{\upsilon, \epsilon\}$; where υ is the set of nodes and ϵ is the set of (undirected) edges. Each node $v \in \upsilon$ has three possible states: susceptible (S), infected (I), and recovered (R). We assume a time slotted system. Nodes change their states at the beginning of each time slot, and the state of node υ in time slot t is denoted by $X_v(t)$.

Initially, all nodes are in state S except node v^* which is in state I and is the information source. At the beginning of each time slot, each infected node infects each of its susceptible neighbors with probability q; independent of other nodes, i.e., a susceptible node is infected with probability $1-(1-q)^n$ if it has n infected neighbors. Each infected node recovers with probability p, i.e., its state changes from I to R with probability p: In addition, we assume a recovered node cannot be infected again. Since whether a node gets infected only depends on the states of its neighbors and whether a node becomes a recovered node only depends on its own state in the previous time slot, the infection process can be modeled as a discrete time Markov chain $X(t)$ where $X(t) = \{X_v(t), v \in \upsilon\}$ is the states of all the nodes at time slot t: The initial state of this Markov chain is $X_v(0) = S$ for $v \neq v^*$ and $X_{v^*}(0) = I$.

2.5.2 Diffusion process:

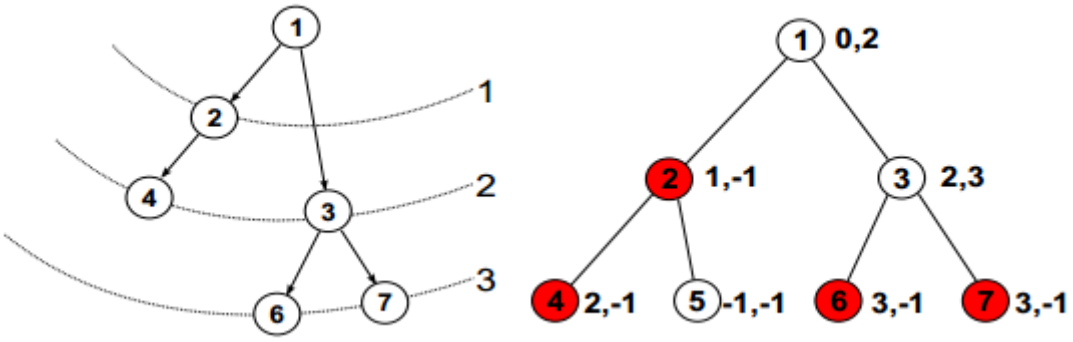

Figure 7: An Example of Information Propagation

We assume $X(t)$ is not fully observable since we cannot distinguish susceptible nodes and recovered ones. So at time t, we observe $Y = \{Y_v, v \in v\}$; such that:

$$Y_v = \begin{cases} 0 & \text{if v is in state I} \\ 1 & \text{if v is in state S or R} \end{cases} \tag{3}$$

The information source detection problem is to identify v, given the graph G and Y; where t is an unknown parameter. Figure 1 is an example of the infection process. The left figure shows the information propagation over time. The nodes on each dotted line are the nodes which are infected at that time slot, and the arrows indicate where the infection comes from (e.g., node 4 is infected by node 2). The figure on the right is the network we observe, where the shaded nodes are infected nodes and others are susceptible or recovered nodes. The pair of numbers next to each node are the corresponding infection time and recovery time. For example, node 3 was infected at time slot 2 and recovered at time slot 3. -1 indicates that the infection or recovery has yet occurred. Note that these two pieces of information are not available to us, and we include them in the figure to illustrate the infection and recovery processes. If we observe the network at the end of time slot 3; then the snapshot of the network is Y = {0; 1; 0; 1; 0; 1; 1}; where the states are ordered according to the indices of the nodes.

3 Gradient Maximum Likelihood Algorithm

This section describes a method that uses a limited number of nodes to act as observers, and observers report the time at which information travels to them. When the observers are selected, nodes that have a longer propagation time are ignored. The method of finding the source is based on the observer's likelihood gradient. This method is called the gradient maximum likelihood algorithm (GMLA). We performed numerical tests on synthetic networks and explore the relationship between key parameter and the algorithm performance.

3.1 Model

We simulate the spread through the network using discrete Susceptible-Infected (SI) model like Figure 8. In this model, each node can be in one of two states:

susceptible or infected. At t = 0 only one random node is infected. We called this node the true source. At each subsequent time step each infected node has a chance to pass the information to its neighbor. The number of chances per time step is equal to the number of neighbors and for each neighbor the probability of success P is the same. The parameter P is called the infection rate. In this article, P = 0.5.

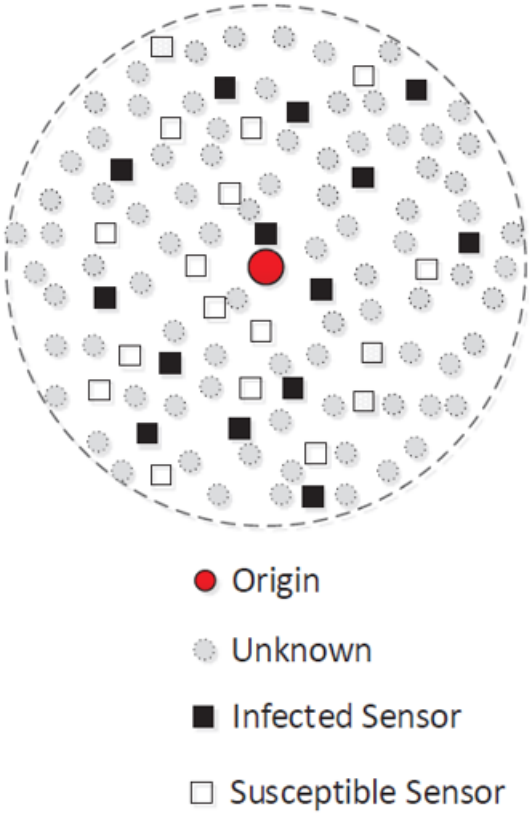

Figure 8: SI Model

We use the following three indicators to measure the performance of the algorithm:

The accuracy of a single realization is $a_i = 1/V_{top}$, if s* $\in V_{top}$ or $a_i = 0$ otherwise, where s* is the true source and V_{top} is a group of nodes with the highest score (top scorers). The total accuracy a is an average of many realizations a_i, therefore a $\in$ [0, 1]. This measure takes into account the fact that there might be more than one node with the highest score.

The rank is the position of the true source on the node list sorted in descending order by the score. In other words this measure shows how many nodes, according to the algorithm, is a better candidate for a source than the true source. The rank takes into account the fact that the algorithm, which is very poor in pointing out the source exactly (low accuracy), can be very good at pointing out a small group of nodes among which is the source.

The distance error is the number of edges between the true source and a node designated as the source by the algorithm. If $|V_{top}| > 1$, which means that the algorithm found more than one candidate for the source, the distance error is computed as a mean shortest path distance between the real source and the top scorers.

3.2 Algorithm

The algorithm is called the Gradient Maximum Likelihood Algorithm(GMLA). In the network, some nodes are selected to be observers, and the observers report the time when their neighbors received the information. To be more general, we do not require that the data received by the observer include the identity of the communicator. Then use the time reported by all observers to calculate the probability that each node is the source, which we call score. We assume that information is always propagated along the shortest path, so we use a breadth-first search tree. We also assume that the time of propagation of each edge is an independent Gaussian variable whose mean and variance are known. This is the preliminary algorithm with complexity above $O(N^3)$.

Due to the high complexity, it is difficult to apply to the actual situation and it needs to be improved. On the one hand, because of too many observers, the far observers′ contribution to the results are small, but the computational costs are particularly high. Therefore, we only use the most recent K_0 observers, which can greatly shorten the calculation time. On the other hand, when calculating the nodes′ scores, all the observers' neighbors are calculated, so the complexity is too high. Therefore, we first calculate one observer′ nearest neighbors′ scores, then chooses the highest score neighbor. Then jump to this node, calculate the scores of its nearest neighbors and repeat the cycle. The whole process is like a gradient descent and continues until all neighbors′ scores are lower than the observer.

3.2.1 Visualization

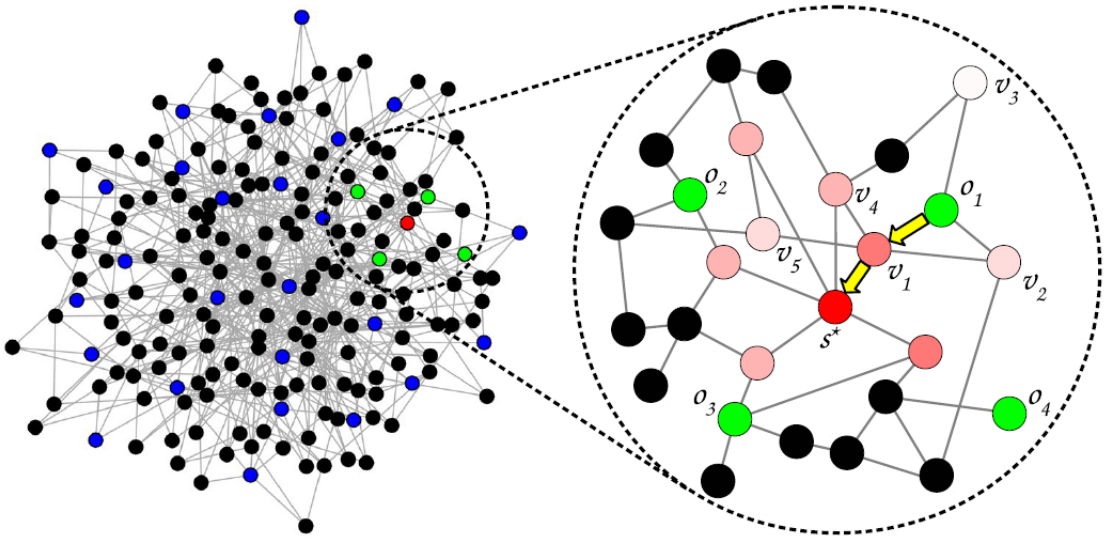

Figure 9: The Visualization Of GMLA

As we can see in Figure 9, the left picture presents the whole graph. The red node is the true source. Green nodes mark the K_0 nearest observers with the smallest time delays (in this plot $K_0 = 4$). The rest of the observers in the network are highlighted in blue. The picture on the right is a zoom of a small area around the nearest observers. In this picture the color corresponds to the score (the likelihood of being the source) of the node (except for the observers which are green). The higher the score of the node is, the darker red is its color. At the beginning the algorithm computes the scores for the neighbors of the nearest observer (in this plot

the observer one is o_1 and its neighbors are v_1, v_2, and v_3). Afterwards GMLA selects the neighbor with the highest score (v_1 in this case) and starts computing the scores for its neighbors (o_1, v_4, v_5 and s*). During this step there is no need for estimating the likelihood for the node v_2 since it was done in the previous step. All the scores which are computed are stored in the list. Since s* has the highest score among the neighbors of v_1, in the next step GMLA will compute the scores for it neighbors. None of the neighbors of s* has higher score than s*, therefore the algorithm stops here. The node s* is the source according to GMLA because it has the highest score from all tested (suspected) nodes. The nodes not visited by the algorithm are black since their scores are undefined (their scores are not computed).

3.2.2 Complexity

The number of suspicious nodes $N_0 = |V_s|$ depends mainly on the size of the network and the average degree $< k >$. We know that $N_0 \sim klog(N)$. It is worth noting that this algorithm does not guarantee that the real source s* will be chosen for the score calculation, so the accuracy may not be very high.

Using the symbols K_0 and N_0, we redefine the time complexity of GMLA as $O(N_0(K_0^3 + N^2))$ in the worst case. Assuming $N_0 \sim log(N)$ and $K_0 << N$, so that the complexity can be further simplified to $O(log(N)N^2)$.

3.2.3 Key Parameter

The number of nearest observers K_0 is a key parameter of GMLA and should be carefully chosen. If K_0 is too small, the accuracy of the algorithm will decrease. On the other hand, a large K_0 increases the calculation time. The optimal number of the nearest observers K_0* is the minimal number of the nearest observers K_0 needed to achieve maximal quality of the spread source localization. In the paragraph below, we explore the relationship between K_0 and the algorithm performance.

3.3 Results

We evaluate the algorithm on synthetic dataset. The dataset is generated by the MMSB model (a generative model that can generate network structures and node attributes simultaneously), which contains 3000 nodes and 17477 edges. We explore the relationship between K_0 and accuracy, rank and distance error.

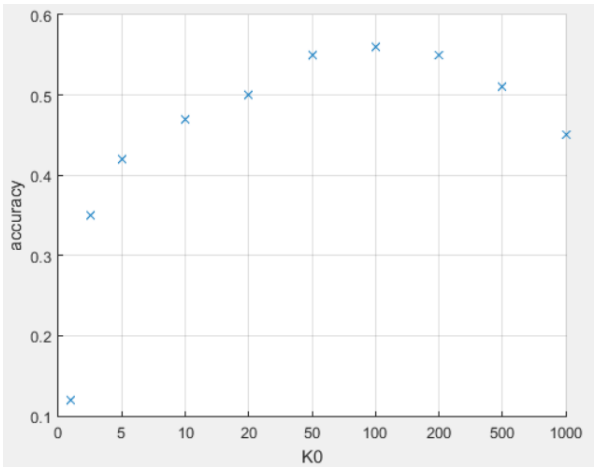

Figure 10: Accuracy

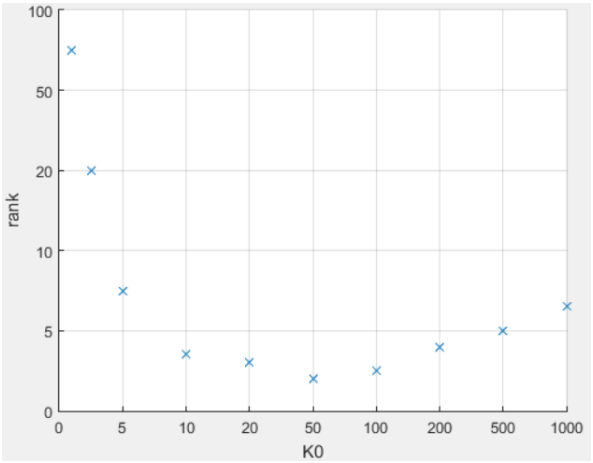

Figure 11: Rank

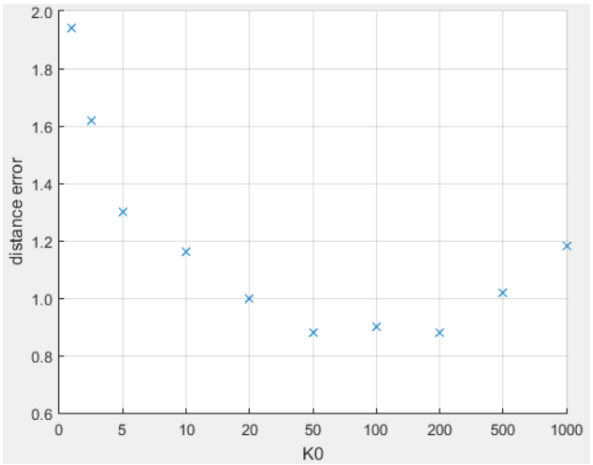

Figure 12: Distance Error

From Figure 10, we can see that as K_0 increases, the accuracy increases first and then decreases, and the maximum value is obtained when $K_0 = 100$. From Figure 11, we can see that with K_0 increasing, the rank decreases first, then increases, and the minimum value is obtained when $K_0 = 50$. From Figure 12, we can see that as K_0 increases, the distance error decreases first and then increases, and the minimum value is obtained when $K_0 = 100$. Thus, in this case, the optimal value range of K_0 is $50 \sim 100$.

3.4 Conclusion

The GMLA algorithm uses a method like gradient descent to calculate the score by reducing the number of observers. The number of suspicious nodes is the logarithmic level of the total number of nodes. Thus, it has a lower complexity and greatly reduces the time required. Although the algorithm does not calculate the neighbors of all observers as other previous algorithm does, we can see from its experimental results that its performance is not bad. The value of K_0 in the algorithm is critical. When it is small, the time required is short, but the performance is not good. When it is big, it performs well, but takes a long time. If it is too big, it will not only take a long time, but also not perform well. Through our verification, in the MMSB model, K_0 takes 50 to 100 is more appropriate. In other networks, the optimal value of K_0 needs experimental exploration.

4 Multiple Sources Locating

The above source locating algorithms are based on an assumption that a network only contains one single source node. However, in fact, information often be distributed from multiple sources. The spread of rumors in social networks, for example, always started from multiple sources. As shown in figure 13, there are two sources in this network. But if we consider it as a singel source locating problem, the yellow node is the most probable solution, which is far from satisfactory. So it is very necessary to solve the problem of multiple sources locating in social network.

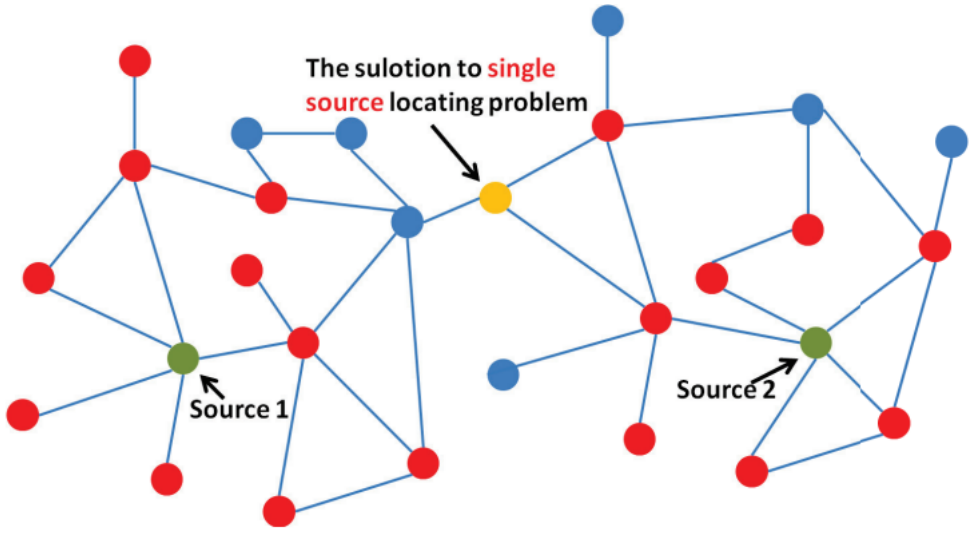

Figure 13: The Problem Of Multiple Sources Locating

4.1 Model

In this problem, we use SIR (Susceptible-Infected-Recovered) model as the propagation model of network. The difference between SI model and SIR model is that, the SIR model contains not only susceptible nodes and infected nodes but also recovered nodes. So what is recovered node? Taking rumor propagation for example, besides the fact that people who received the rumor may forwarded it with certain probability, people who has been infected may also find the message is rumor and then delete it with a certain probability, and the latter people are called recovered nodes. For a network consisting of N nodes, in each discrete time-step, infected nodes try to infect their susceptible neighbors with probability p, and these infected nodes can also be recovered with probability q. In addition, all recovered nodes cannot be infected again.

In the SIR propagation model, we cannot only use the available infected nodes, because the information behind the recovered nodes will be neglected. Therefore, in order to solve this problem, we firstly study a reverse propagation method to detect recovered and unobserved infected nodes from susceptible ones ,and we can get an extended infected network. Then, we divide the newly detected network into several parts based on community detection methods. Through this partition, we can transform the multi-source locating problem into several independent single source-locating problems. Eventually we can find out the source of each partition independently based on the algorithm mentioned above.

4.2 Method of Reverse Propagation

When we try to solve the problem of source locating, usually only a few of infected nodes can be observed, so that many useful information such as many recovered and infected nodes are unobserved. Thus, an important task in source locating problem based on the SIR model is to find out more information, and infer the extended infected network through partially observed network.

We use a score-based reverse propagation method to solve this task. Here score means the probability of a node in recovered or infected state. If the probability is larger, the score will be higher. Generally speaking, recovered nodes are often surrounded by infected nodes. That is to say, recovered nodes will obtain higher scores than susceptible nodes. This is because the center of infected nodes will get higher scores than periphery of the infected nodes, while recovered nodes often lie at the center of infected nodes and susceptible nodes usually lie at the periphery of infected nodes.

The input of this algorithm is a social network G=(V,E), which contains a nodes set N, a edge set E, a partially observed infected nodes set $\Gamma \subseteq V$, and a constant basescore, we can optimize the algorithm by adjusting the basescore. The output of this algorithm will be an extended infected nodes set $\Gamma^* \subseteq V$, which contains recovered nodes, observed and unobserved infected nodes, nodes once contact to infected nodes but not be infected. Moreover, we have $\Gamma \subseteq \Gamma^* \subseteq V$.

In this algorithm, we firstly initialize the score of all observed infected nodes to 1 and other nodes with initial score 0. Then we give a label C to every node and initial the labels the same as scores:

$$C_n, S_{Cn} = \begin{cases} 1 & i \in \Gamma \\ 0 & otherwise \end{cases}$$

Then do as the following:

Initialize nodes set $\Gamma^* = \Gamma$
for iter 1 to N_{step} do
  for $n \in \Gamma^*$ do
    for $i \in n_{neighbor}$ do
      update $\Gamma^* = \Gamma^* \cup$ i, $C_i = 1$
      update $S_{Ci} = S_{Ci} + S_{Cn}$, if $C_i = 0$
for $n \in V$ do
  If S_{Cn} >basescore, $\Gamma^* = \Gamma^* \cup$ n
Return extended infected nodes set Γ^*

By performing the algorithm above, we can approximately restore the extended infected network of propagation, which actually presents a new network for source locating analysis.

4.3 Infected Nodes Partitions based on Different Sources

After the score-based reverse propagation above, we get the extended infected network, and what we need to do next is to divide the complex multi-sources locating problem into several single-source locating problems. We assume that all rumor sources propagate the information independently. Then we use the leading eigenvector based partition solution to solve our task.

Generally speaking, a feasible division of infected nodes needs to satisfy two conditions , including sparse edges between different groups and dense edges within

the same group. The conditions can be described as a modified benefit function called modularity:

Q = (number of edges within communities) - (expected number of such edges)

The above equation denotes a function of a particular division of the infected network into different groups, with larger values indicating better divisions. We can maximize it over all possible divisions of the network. However, exhaustive maximization over all possible divisions is computational intractable. The modularity function can be rewritten in matrix terms, which allows us to express the optimization task as a spectral problem in linear algebra.

Another key point in this task is, when partitioning the infected nodes, the number of sources k is required to be known. Since the number of rumor sources is always unknown in practice, we need to estimate the number of sources. Firstly, choose a maximum of k, K. Next, for each k satisfied $1 < k < K$, using the community partition algorithm (Leading eigenvector based method) to partition the extended infected network. Then, we compute the modularity m_k of each partition based on the number of sources, k. Next, we set $k = 2$ and observe the increase of m_k when changing the value of k, $m_k - m_{k-1}$. If there is a significantly increase of m by increasing the value of k, we then modify the value of k to a larger one. Otherwise, we prefer to chose a smaller value for k.

Now we actually have translated multiple source locating problem to several single-source locating problems, and the we can find out the source of each partition independently. Eventually the problem of multiple source locating is solved.

4.4 Evaluation

We evaluate our algorithm on two different networks of different scale, one network consists of 500 nodes and the another consists of 5000 nodes. In addition, the number of sources range form 2 to 4. As figure 14 and figure 15 showing:

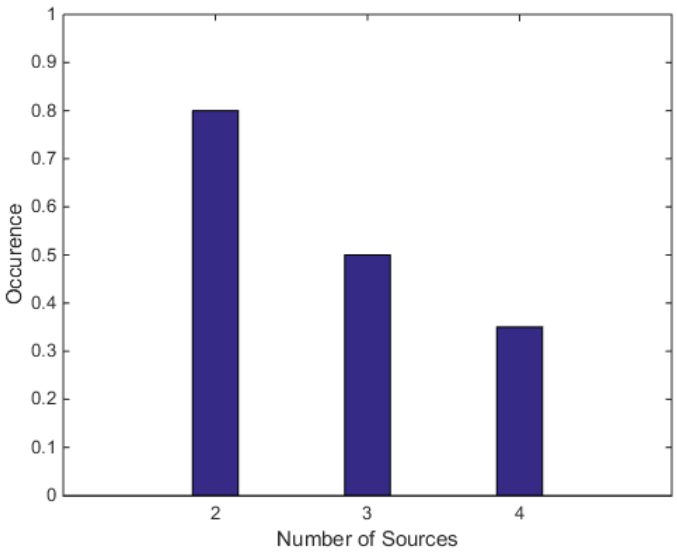

Figure 14: Accuracy Of Source Locating In Network Of 500 Nodes

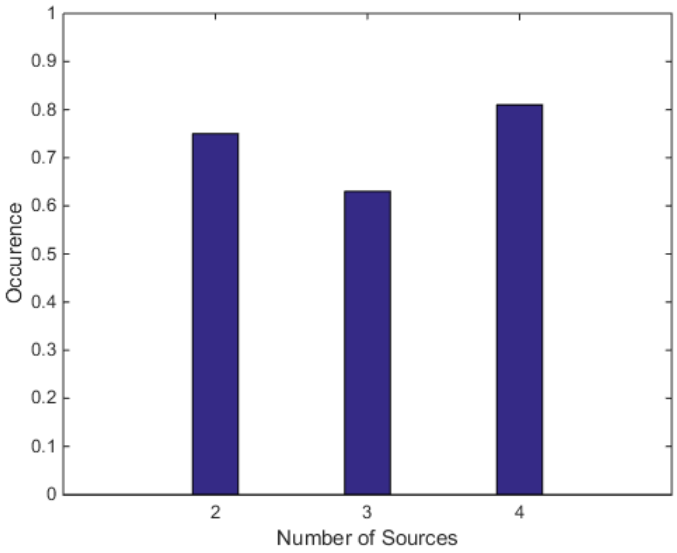

Figure 15: Accuracy Of Source Locating In Network Of 5000 Nodes

From the results above we can find that, the overall accuracy of large-scale network is higher than the small-scale network. When the number of nodes is relatively small, the accuracy will decrease with the increase of number of sources. But in relatively large-scale network, the number of sources within a small range seems to have no effect on the accuracy of source locating.

5 Conclusion

In this paper, we introduce A Sample Path Based Approach to solve the source locating problem in social networks. Furthermore, we propose Gradient Maximum Likelihood Algorithm to reduce the complexity. Finally, we do some research on multiple sources locating problem.