Vamsi Mokkapati

TA: Sharath Gopal

CS 35L Assignment 10 Report

3 June 2016

Rumor Has It…An Algorithm Could Scope Out Gossip!

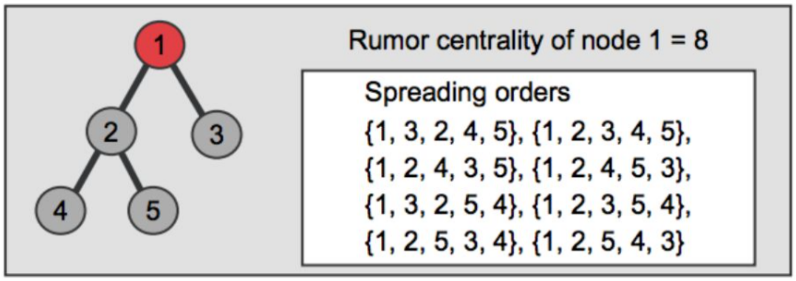
One of the networking problems that is has been intriguing researchers recently is finding accurate methods to tell them where a rumor initially originated from. Although this question sounds deceptively simple, especially when we're given information on the time at which individuals heard the rumor; the answer would then be simply the person who heard the rumor at the earliest time. However, finding such timing information has been observed to be exceedingly difficult; therefore, the parameters of this problem have been specifically set to finding the probable origin of a rumor given a graph, with nodes in the graph representing individuals, and edge between node A and node B representing the fact that the rumor has spread from A to B.

In this paper, I will be analyzing at a high level two algorithms that help find the origin node given a complex graph: the first algorithm, developed by Devavrat Shah and Tauhid Zaman of MIT, focuses on finding a value called the rumor centrality parameter (RCP) for each node given a tree graph in order to help find the origin node, while the second method, developed by Lei Ying and Kai Zhu, uses something known as the Short-Fat Tree (SFT) algorithm and works on a much wider range of graphs to perform the same function.

When analyzing the first method by Shah and Zaman, it has to be kept in mind that it only works under a specific set of conditions. First, the graph being analyzed has to be a tree, which means that any two vertices have only one path between them, and implies that there are no cycles present. Also, it has to be kept in mind that this method assumes that all branches have an equal likelihood to be set; in terms of rumors, this means there's an equal likelihood of the rumor spreading from one node to the next.

The key method used in Shah and Zaman's algorithm is to find the RCP at each node; the node with the highest RCP value is most likely to be the origin. Take Figure 1 as an example below:

(Color online) Example of rumor centrality calculation for a 5 node network.



Note. The rumor centrality of node 1 is 8 because there are 8 spreading orders that it can originate, which are shown in the figure.

Figure 1

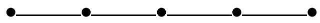
In essence, the RCP is nothing but the total number of spreading orders that can be present from that node down to the leaves of the tree. As illustrated in Figure 1, it is clear that there are 8 different permutations in which the rumor could be spread along the tree from node 1; similarly, we can find that the RCP at node 2 is 2, and the RCP at nodes 3, 4, and 5 is 1. Therefore, since the RCP has the highest value at node 1, we know that it is the most probable node at which the rumor originated from. If we take each node to be a person, and each edge to be a representation of one person telling a rumor to another, then we can use this very basic algorithm to obtain an educated guess about which person started the rumor.

Now given a complex graph of hundreds of thousands of nodes, how can we quickly find the RCP values at each node, and see which node has the highest RCP? In their research paper, Shah and Zaman have derived the following formula to do so:

$$R(u, G) = \frac{|V|!}{\prod_{w \in V} T_w^u}$$

In the formula above, V represents the set of nodes, while T_w is the set of subtrees from that node. Therefore, the formula states that the RCP at a given node is the factorial of the total number of nodes in the subtree at the starting node divided by the product of the number of nodes in each of those nodes' subtrees. For example, in Figure 1, if we wanted to find the RCP at node 1, |V| = 5, and R(*u*, G) = 5!/(5*3*1*1*1) = 8, and so on.

One note that is important to be made from this formula and the definition of RCP is that this algorithmic method will NOT be useful on graphs with exceedingly easy complexities, such as a linear network with each node only having an edge to one other node, such as the following:



In the simple graph above, it is quickly seen through inspection and using the formula above that the RCP at each node has the same value (in this case, the RCP for all the nodes is 5!, or 120), and that therefore no reasonable estimate can be arrived at to know which node was the origin of the rumor.

Another interesting note about Shah and Zaman's method is that using the RCP to find the origin node is only an estimate, and certainly not a guarantee that the final node is indeed the origin of the rumor. However, after doing extensive statistical analysis in their research paper, both researchers found that the probability of the true source being further than *k* hops away from the source estimated by RCP calculations exponentially decreases. Therefore, this algorithm is useful in that we can be fairly certain that the true origin node is always going to be in the general vicinity of the estimated origin node.

Whereas the RCP method described earlier had some constraints in the form of the type of graphs it could work on, and the fact that all branches automatically have an equal likelihood of being set, Ying and Zhu's Short-Fat Tree (SFT) Algorithm has considerably fewer restrictions, and is generally considered to be an improvement on Shah and Zaman's research, since it is based on it to an extent.

While the formers' approach only works on graphs that are regular trees, the SFT algorithm works on all Erdős-Rényi random graphs, which includes trees and a much broader range of other more complex graphs. Also, the SFT algorithm is designed such that it takes into account the probability of one node being "infected" by another node for all nodes (an infected node is one that has heard the rumor, in this case), whereas the earlier algorithm assumes equal infection probability for all nodes; it does this using something known as a weighted boundary node degree (WBND) measure, which is calculated as follows:

$$\sum_{(u,w)\in\mathcal{F}'_v} |\log(1-q_{uw})|,$$

where q_{uw} is the infection probability for each pair of nodes.

The SFT algorithm is given its name because it identifies the source node as the one that has the minimum depth while having the most leaf nodes (therefore being the parent of the "shortest, fattest tree"). Figure 2 below illustrates an example of its iteration on a graph; note how this graph wouldn't work on the earlier algorithm since it is not a tree, and contains cycles in it:

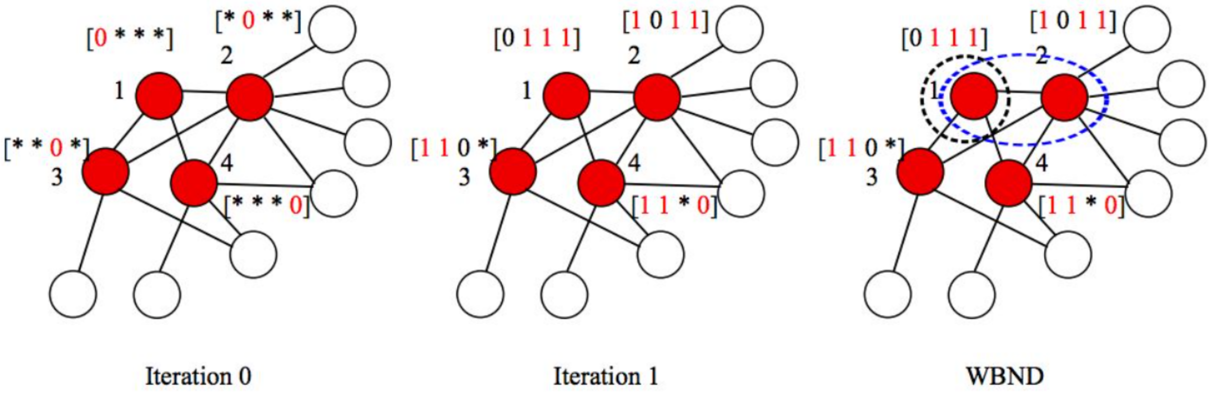
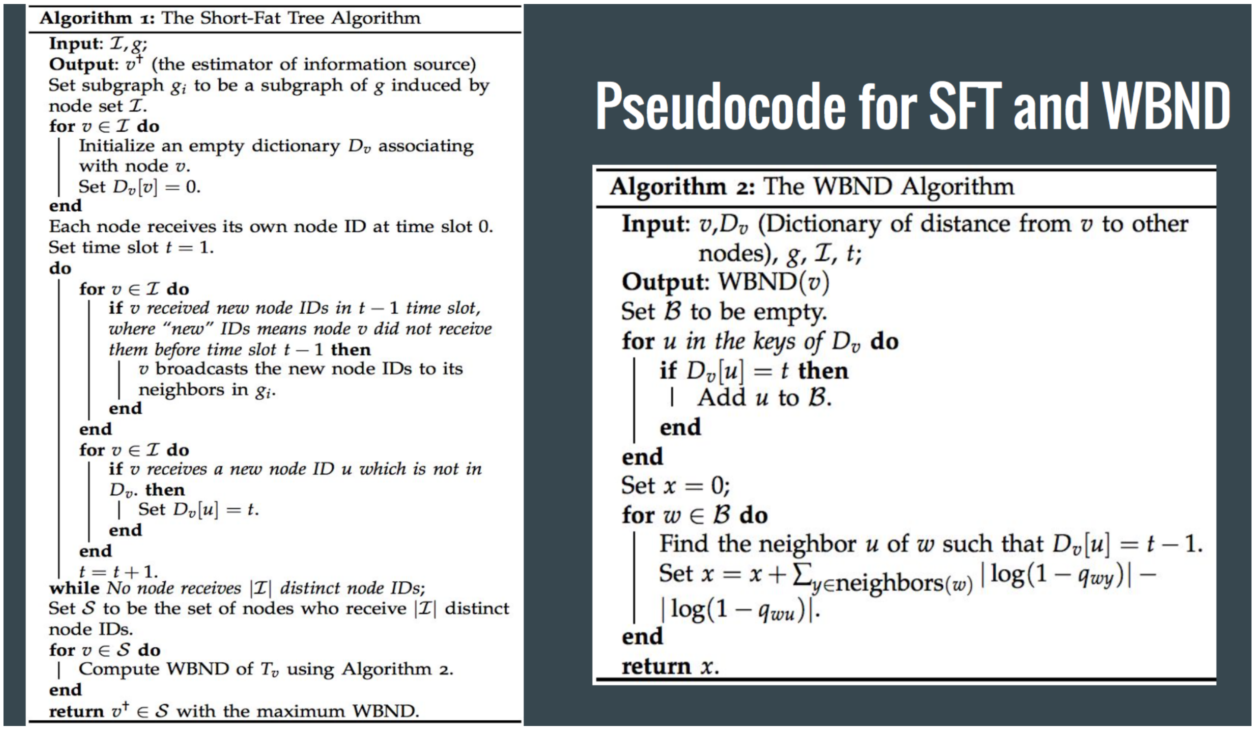


Figure 2

In Figure 2 above, the red dots represent infected nodes. In Iteration 0, each infected node has an array initialized to the size of how many infected nodes there are in the graph; the number 0 at each node's respective number illustrates that the infected node is 0 steps away from itself. Iteration 1 has each node recognizing all the infected nodes it's connected to; for instance, every node that node 1 is connected to is infected; therefore, its array is [0 1 1 1]. However, node 3 is not connected to node 4, but connected to infected nodes 1 and 2, so its array is [1 1 0 *]. In the third iteration, the program uses the WBND measure at each node to find out that the value is highest at node 1 in this case, assuming the weights of all edges are equal. In a real-life scenario where the weights of all edges are NOT equal, it is possible that node 2 could be the source node instead of node 1, hence the dotted blue ellipse around nodes 1 and 2.

Figure 3 shows the pseudocode for the SFT algorithm, and the WBND algorithm algorithm which it calls. Both the algorithms and their implementations are shown at a deeper level below:



Pseudocode for SFT and WBND

```
Algorithm 1: The Short-Fat Tree Algorithm
  Input: $\mathcal{I}, g$;
  Output: $v^\dagger$ (the estimator of information source)
  Set subgraph $g_i$ to be a subgraph of $g$ induced by node set $\mathcal{I}$.
  for $v \in \mathcal{I}$ do
      Initialize an empty dictionary $D_v$ associating with node $v$.
      Set $D_v[v] = 0$.
  end
  Each node receives its own node ID at time slot 0.
  Set time slot $t = 1$.
  do
      for $v \in \mathcal{I}$ do
          if $v$ received new node IDs in $t - 1$ time slot, where "new" IDs means node $v$ did not receive them before time slot $t - 1$ then
              $v$ broadcasts the new node IDs to its neighbors in $g_i$.
          end
      end
      for $v \in \mathcal{I}$ do
          if $v$ receives a new node ID $u$ which is not in $D_v$. then
              Set $D_v[u] = t$.
          end
      end
      $t = t + 1$.
  while No node receives $|\mathcal{I}|$ distinct node IDs;
  Set $\mathcal{S}$ to be the set of nodes who receive $|\mathcal{I}|$ distinct node IDs.
  for $v \in \mathcal{S}$ do
      Compute WBND of $T_v$ using Algorithm 2.
  end
  return $v^\dagger \in \mathcal{S}$ with the maximum WBND.
```

```
Algorithm 2: The WBND Algorithm
  Input: $v$,$D_v$ (Dictionary of distance from $v$ to other nodes), $g$, $\mathcal{I}$, $t$;
  Output: WBND($v$)
  Set $\mathcal{B}$ to be empty.
  for $u$ in the keys of $D_v$ do
      if $D_v[u] = t$ then
          Add $u$ to $\mathcal{B}$.
      end
  end
  Set $x = 0$;
  for $w \in \mathcal{B}$ do
      Find the neighbor $u$ of $w$ such that $D_v[u] = t - 1$.
      Set $x = x + \sum_{y \in \text{neighbors}(w)} |\log(1 - q_{wy})| - |\log(1 - q_{wu})|$.
  end
  return $x$.
```

Figure 3

After analyzing both the RCP and the SFT algorithms and comparing the two source localization techniques, it is clear that the SFT algorithm has a wider range of applications due to its applicability to a much wider group of graphs and its ability to take weighted boundary node degree into account. And it should also be noted that this wide range of applications is not strictly confined to the problem of rumor origination. These algorithms are very useful in today's world to help solve a large variety of problems, including finding the origins of computer viruses, locating the origin of news sources to analyze news credibility, and even the area of epidemiology, which is the study of the incidence and origin of diseases. On a more recreational note, these techniques could also be used to find the origin of fashion trends, and even the origin of certain memes used on the internet. This broad range of use for these algorithms makes it useful to continue research in this area. Currently, research is being done on how to find the origin node when we don't have full information on how much the rumor has spread.

Works Cited

1. Shah, Devavrat, and Tauhid Zaman. "Finding Rumor Sources on Random Trees." Operations Research (2016): n. pag. Web. 5 May 2016.
2. Woo, Marcus. "Rumor Has It An Algorithm Could Scope Out Gossip."Inside Science. Inside Science, 11 Mar. 2016. Web. 05 May 2016. <https://www.insidescience.org/content/rumor-has-it-algorithm-could-scope-out-gossip/3756>.
3. Zhu, Kai, and Lei Ying. "Source Localization in Networks: Trees and Beyond." (n.d.): n. pag. Web. 5 May 2016.