Final Report
Influence Identification and Maximization on Independent Cascade Model

HUANG Hongru, MENG Jingfan, WANG Xuyao

1. Introduction

Social network — modeled as a graph of relationships and interactions within a group of individuals — plays a fundamental role as a medium for the spread of information, ideas, and influence among its members. One of the primary phenomena that is extensively studied in social networks is the spread of contagion, where infection starts from one or multiple sources and spread to a larger scope of vertexes. Many interesting patterns, such as the existence of a critical count of seeds and a sharp phase transition after which almost the entire network is affected by infection, have been unveiled and testifies its significance not only in network stability, epidemics and privacy protection, but also in various field of physics and bio-informatics.

A motivating application for the study of influence spread is active propagation of valuable information, often seen in cases of viral marketing strategies, in which a fraction of customers are provided with free copies of a product, and the retailer desires the number of adoptions triggered by such trials to be maximized. Here, the problem is to select a fixed number of seed nodes that will attain maximal influence power. The problem also has important applications beyond social graphs, such as placing sensors in water distribution networks for detecting contamination.

Our work is mainly inspired by the framework in [1] on the optimal percolation. Its main idea is to calculate the probability of a node belonging to the giant component in message passing formulas, and to minimize the leading eigenvalue of the Jacobian matrix at its zero solution to make the zero solution stable. [1] also shows that the optimal percolation problem can be converted to finding the minimal set of immunized nodes to protect the network from infection.

In our work, we are going to study the complementary problem of finding the minimal set of nodes to infect as many other nodes as possible. We generally follow the framework in [1], but with message passing formulas specific to this problem, and derivations a little more complex than the original problem.

2. Problem Formulation

The problem of influence maximization has been extensively studied, yet we put a formal formulation of the problem here for the completeness and understandability of our report.

The social network is modeled by a (directed) graph G with a weight p_{ij} associated to each edge in G. When the network is undirected, we have $p_{ij} = p_{ji}$.

The infection starts from an arbitrary set of nodes named seeds, which are infected initially by external powers and spread to other parts of the network through links between nodes. Each edge $e = (i, j)$ is associated with a probability p_{ij} for successful transmission, and the outcomes on different edges are independent from each other. When a node becomes infected, it maintains the infected state and can in turn infect other nodes through links. The spread of infection stops when no node gets newly infected after all edges have been checked, and the target to optimize is the number of infected nodes in the end.

The problem is, to find the most influential set of seeds with at most K nodes to maximize the expected number of infected nodes after spreading. Formally, this is to find

$$S^* = \arg \max_{S:|S| \le K} \sum_{i \in V} u_i \tag{1}$$

This problem has been proved to be NP-hard by [2], and many works seeking approximate solutions have appeared since then, such as utilizing the sub-modular property by greedily selecting the node with maximum marginal influence. We would try to approach this problem using the idea of collective influence, hoping to get a more efficient and effective algorithm.

The following is a list of important symbols that will be used in this report.

Table 1. List of symbols and paremeters

Symbol	Description
A_{ij}	Term (i, j) in adjacency matrix of G.
$\mathbf{B}_{ij}$	Jacobian matrix on edge (i, j) as a simplification of $\mathbf{M}$ in (19).
$CI_l(i)$	Collective influence of node i used as metric for seed selection.

d_k	Degree of node k.
$E_{t,ij}$	Event of node j successfully infected by i.
$E_{u,ij}$	Event of node i being infected in $G \setminus j$.
$E_{v,ij}$	Event of node i belonging to the giant component of $I_S(G) \setminus j$.
$G(V,E)$	Graph representation of a social network.
	This is where spreading of infection happens.
i, j, k	Nodes in V.
$I_S(G)$	Set of infected nodes given seed set S.
K	Maximum number of nodes in seed set.
l	Truncating length of local paths in CI calculation.
$\mathbf{M}$	Jacobian matrix of equations (11)(13) at zero solution.
n_i	Indicator of whether i is selected as seed.
	$n_i = 1$ if $i \in S$, and $n_i = 0$ otherwise.
p_{ij}	Probability of successful transmission on edge (i, j).
S	Set of seeds which initiates infection.
u_i	Probability of node i being infected.
u_{ij}	Probability of node i being infected in $G \setminus j$.
	This precludes the case where i is infected by j.
v_i	Probability of i belonging to $I_S(G)$.
	Giant component of infected nodes appear when not all v_i are zero.
v_{ij}	Probability of i belonging to $I_S(G) \setminus j$.
	Preclude the case where i belongs to the giant component because of j.
$\mathbf{w}_l$	The l-th order vector in power iteration.
w_{ij}	$P(E_{v,ij} \mid E_{u,ij})$.
y_{ij}	$P(E_{v,ij} \mid \bar{E}_{t,ij})$. Probability of i still belonging to giant component
	even if no successful infection observed.
z_{ij}	$P(E_{v,ij} \mid \bar{E}_{u,ij})$. w_{ij} and z_{ij} are variables
	describing the joint probability of these two events.
$\Gamma(i)$	Set of neighbors of node i.
ε	Accuracy parameter of estimating u_{ij}.
λ	Leading eigenvalue (spectrum radius) of $\mathbf{M}$.

3. Message Passing Formula and Eigenvalue Optimization

3.1. Estimation of probability of infection

The following analysis is based on the framework in [1], which is on the optimal percolation on a graph. Its main idea is to calculate the probability of a node belonging to the giant component in message passing formulas, and to minimize the leading eigenvalue of the Jacobian matrix at its zero solution to make this solution stable.

Now, we start by considering the possibility that one vertex belongs to the giant component of infected nodes. Let v_i denote the possibility that vertex i belong to a giant component of infected nodes $I_S(G)$ in graph G, and v_{ij} be that probability of considering j's infection to i but not considering j is a member of giant component. In other words, v_{ij} is the probability that i is infected in G and i belongs to the giant component in $I_S(G) \setminus j$. Let u_i be the probability that i get infected in G, and u_{ij} be that probability in $G \setminus j$.

Let n_i indicate whether vertex i is selected as seed of infection: $n_i = 1$ if vertex i is seed and $n_i = 0$ otherwise. First, we write the equation between u_{ij} (message passing formulas).

$$u_{ij} = n_i + (1-n_i)\left[1 - \prod_{k\in\Gamma(i)\setminus j}(1-p_{ki}u_{ki})\right] = 1-(1-n_i)\prod_{k\in\Gamma(i)\setminus j}(1-p_{ki}u_{ki}) \tag{2}$$

In this equation, p_{ki} is the success probability of transmission from k to i. $\Gamma(i)$ is the set of neighbors of i. If $n_i = 1$, u_{ij} is always 1, and otherwise, u_{ij} is the probability of at least one neighbor being infected. These probabilities are multiplied under the assumption of local tree structure , in which the infection of neighbors are independent.

And the u_i is calculated without removing j.

$$u_i = 1-(1-n_i)\prod_{k\in\Gamma(i)}(1-p_{ki}u_{ki}) \tag{3}$$

With equations (3)(2), given seed selection $\mathbf{n}$, the probability of infection of all nodes $\mathbf{u}$ can be calculated by iterating(2). The solution would be a stable fixed point of these equations, yet we do not know the exact solution.

In reality, calculating u_{ij} using (2) may take too long to converge. Therefore, an alternative method is required to quickly estimate u_{ij}.

The alternative method is to search from i in the local tree structure of graph $G \setminus j$, stopping at seed nodes or when the joint probability of successful transmission on the path from root is less than a threshold ε. In formal expression, stop at node k when

$$n_k = 1 \quad or \quad \prod_{e \in p(i,k)} p_e < \varepsilon$$

This method takes advantage of the fact that infection rarely spread through long paths on usual conditions, so it is legitimate to only consider the local structure near i.

In the resultant tree T_{ij}, remove all branches not containing seed nodes as leaves. Set the probability of infection x_k to 1 for all seed leaves k. Then, recursively compute x_k for all nodes k with all children $s \in \Gamma'(k)$ having known x_s.

$$x_k = 1 - \prod_{s \in \Gamma'(k)} (1 - x_s) \tag{4}$$

Iterate with this formula until the infection probability of root node i is computed. The accuracy of this estimation is guaranteed by the formula sub-linear property: if $\prod_{e \in p(i,k)} p_e < \varepsilon$,

$$P(E_{u,ij}|E_{u,kj}) - P(E_{u,ij}|\bar{E}_{u,kj}) < \varepsilon \tag{5}$$

. That is, whether node k is infected will only have trivial influence on the probability that node i is infected because the connection between them is too weak.

3.2. Derivation of Message Passing Formulas

After u_i are calculated, we may try to write the equations of v_{ij} under the message passing framework. However, the joint distribution of u_{ij} and v_{ij} must be modeled explicitly before this.

Let the event $E_{u,ij}$ happen when node i is infected in $G \setminus j$, and $E_{v,ij}$ happen when node i belongs to the giant component of $I_S(G) \setminus j$. Let $w_{ij} = P(E_{v,ij}|E_{u,ij})$, and $z_{ij} = P(E_{v,ij}|\bar{E}_{u,ij})$. Then,

$$v_{ij} = u_{ij}w_{ij} + (1 - u_{ij})z_{ij} \tag{6}$$

The event $E_{v,ij}E_{u,ij}$ happens when i is a seed node or infected by at least one neighbor in $G \setminus j$, and at least one neighbor except j belong to $I_S(G) \setminus i$. Let $E_{t,ij}$ happen when node j is infected by i, with probability $P(E_{t,ij}) = u_{ij}p_{ij}$. Define

$$y_{ij} = P(E_{v,ij}|\bar{E}_{t,ij}) = \frac{u_{ij}(1 - p_{ij})w_{ij} + (1 - u_{ij})z_{ij}}{1 - u_{ij}p_{ij}} \tag{7}$$

as the probability i belongs to giant component in $I_S(G) \setminus v$ given that j is not infected by i.

For all $k \in \Gamma(i) \setminus j$, let $I_{t,ki}$ indicate whether $E_{t,ki}$ happens. Let set $S_i = \{(I_{t,ki})_k : \sum_k I_{t,ki} > 0\}$ be all possible assignments of $I_{t,ki}$ such that node i is infected by at least one neighbor.

$$\begin{aligned}
u_{ij}w_{ij} &= n_i \left[1 - \prod_{k\in\Gamma(i)\setminus j} (1 - v_{ki})\right] \\
&\quad + (1-n_i) \sum_{(I_{t,ki})_k \in S_i} \prod_{k\in\Gamma(i)\setminus j} \left[I_{t,ki} p_{ki} u_{ki} + (1 - I_{t,ki})(1 - p_{ki} u_{ki})\right] \\
&\quad \left[1 - \prod_{k\in\Gamma(i)\setminus j} (1 - I_{t,ki} w_{ki} - (1 - I_{t,ki}) y_{ki})\right] \\
&= 1 - \prod_{k\in\Gamma(i)\setminus j} (1 - v_{ki}) - (1 - u_{ij}) \left[1 - \prod_{k\in\Gamma(i)\setminus j} (1 - y_{ki})\right]
\end{aligned} \tag{8}$$

The last equation can be interpreted as: $E_{v,ij}E_{u,ij}$ does not happen when none of the neighbors in $\Gamma(i) \setminus j$ belongs to $I_S(G) \setminus i$, or node i is not infected in $G \setminus j$ and at least one neighbor belongs to $I_S(G) \setminus i$ (this condition ensures that the 2 alternatives are mutually exclusive). Or by the following formula

$$\sum_{(I_k)_{k=1:n} \in \{0,1\}^n} \prod_k [I_k a_k + (1 - I_k) b_k] = \prod_k (a_k + b_k) \tag{9}$$

, where (I_k) corresponds to a term in the expanded summation of taking one term a_k or b_k from each factor k.

$E_{v,ij}\bar{E}_{u,ij}$ happens only when i is infected directly by j, and at least one neighbor except j belong to $I_S(G) \setminus i$.

$$\begin{aligned}
(1 - u_{ij}) z_{ij} &= p_{ji} u_{ji} (1 - n_i) \prod_{k\in\Gamma(i)\setminus j} (1 - p_{ki} u_{ki}) \left[1 - \prod_{k\in\Gamma(i)\setminus j} (1 - y_{ki})\right] \\
&= (u_i - u_{ij}) \left[1 - \prod_{k\in\Gamma(i)\setminus j} (1 - y_{ki})\right]
\end{aligned} \tag{10}$$

v_{ij} is calculated by combining(6)(8)(10).

$$v_{ij} = 1 - \prod_{k\in\Gamma(i)\setminus j} (1 - v_{ki}) - (1 - u_{ij}) \left[1 - \prod_{k\in\Gamma(i)\setminus j} (1 - y_{ki})\right] \tag{11}$$

Likewise, the probability that node i is in the giant component of $I_S(G)$ is calculated without removing j from $\Gamma(i)$.

$$v_i = 1 - \prod_{k\in\Gamma(i)} (1 - v_{ki}) - (1 - u_i)\left[1 - \prod_{k\in\Gamma(i)} (1 - y_{ki})\right] \tag{12}$$

Also we can get the formula of y_{ij} by combining (7)(8)(10).

$$y_{ij} = \frac{1-p_{ij}}{1-p_{ij}u_{ij}}\left[1 - \prod_{k\in\Gamma(i)\backslash j} (1 - v_{ki}) - (1 - \frac{u_i - p_{ij}u_{ij}}{1-p_{ij}})\left[1 - \prod_{k\in\Gamma(i)\backslash j} (1 - y_{ki})\right]\right] \tag{13}$$

3.3. Condition of Stable Zero Solution and Jacobian Matrix

What we care about is the emergence of giant components in $I_S(G)$, indicated by non-zero values of v_i. By (12), $v_i = 0$ if all independent variables v_{ij} and y_{ij} are zero, which is also a solution of equations (11)(13) (zero solution).

As the variables v_{ij} and y_{ij} depend on each other in (11)(13). It shall be computed by iterating from a random initial value until convergence. A necessary condition of converging to one solution is that the absolute value of leading eigenvalue of the Jacobian matrix at this solution must be smaller than 1.

Now we will justify this observation. First, denote the independent variables as $\mathbf{v}$, so the iteration process can be regarded as evaluating $\mathbf{v}$ over a specific function $f(\mathbf{v})$. Then the result is assigned to $\mathbf{v}$ until $\mathbf{v}$=f($\mathbf{v}$).

At a small neighborhood near $\mathbf{v} = \boldsymbol{\vartheta}$, the function f can be approximated linearly by its Jacobian matrix $\mathbf{M}$, whose element is $M_{ij} = \frac{\partial f_i}{\partial v_j}|_{\forall i, v_i=0}$. For a small error vector $\boldsymbol{\varepsilon}_0$,

$$f(\boldsymbol{\varepsilon}_0) = \mathbf{M}\boldsymbol{\varepsilon}_\mathbf{0} + o(\boldsymbol{\varepsilon}_0) \tag{14}$$

Repeating l times, the inner product of l-th error vector is

$$\langle \boldsymbol{\varepsilon}_l | \boldsymbol{\varepsilon}_l \rangle = (1 + o(1))\boldsymbol{\varepsilon}_0^T (\mathbf{M}^T\mathbf{M})^l \boldsymbol{\varepsilon}_0 \tag{15}$$

When $l \to \infty$, the inner product can be approximated by power of the leading eigenvalue of $\mathbf{M}$. If the leading eigenvalue $\lambda < 1$

$$\lim_{l\to\infty} \langle \boldsymbol{\varepsilon}_l | \boldsymbol{\varepsilon}_l \rangle = \lim_{l\to\infty} \boldsymbol{\varepsilon}_0^T (\mathbf{M}^T\mathbf{M})^l \boldsymbol{\varepsilon}_0 = \lim_{l\to\infty} \lambda^{2l} \langle \boldsymbol{\varepsilon}_0 | \boldsymbol{\varepsilon}_0 \rangle = 0 \tag{16}$$

After enough rounds of iterations, the error vector $\boldsymbol{\varepsilon}$ will eventually come to 0. Therefore, the zero solution of function $f(\mathbf{v})$ is stable.

If the leading eigenvalue $\lambda > 1$, when $\mathbf{M}$ is decomposed in its Jordan canonical form, its Jordan matrix $\mathbf{J}$ contain elements with value $\lambda > 1$ on the diagonal. These diagonal elements will become λ^l in $\mathbf{J}^l$ and will diverge when $l \to \infty$. In this case, $\langle \boldsymbol{\varepsilon}_l | \boldsymbol{\varepsilon}_l \rangle$ will increase continuously until $\boldsymbol{\varepsilon}_l$ leaves the neighborhood in which the derivative is taken to approximate f. As a result, $\boldsymbol{\varepsilon}$ will not approach $\boldsymbol{\vartheta}$ but will end up in another stable solution.

As a conclusion, the solution $\boldsymbol{\vartheta}$ is stable as long as $\lambda < 1$, such that any set of independent variables $\mathbf{v}$ will eventually come to $\boldsymbol{\vartheta}$ after iteration. It is otherwise unstable when $\lambda > 1$, and this is when a small fraction of seed nodes becomes able to set up a round of outbreak to activate the entire network until the giant component of infected nodes emerge.

Inspired by this condition, we now calculate partial derivative of functions (11)(13) at point $\forall i, j, v_{ij} = y_{ij} = 0$ and denote this matrix for ij as $\mathbf{B}_{ij}$.

$$\left.\begin{pmatrix} \frac{\partial v_{ij}}{\partial v_{ki}} & \frac{\partial v_{ij}}{\partial y_{ki}} \\ \frac{\partial y_{ij}}{\partial v_{ki}} & \frac{\partial y_{ij}}{\partial y_{ki}} \end{pmatrix}\right|_{\forall i,j,v_{ij}=y_{ij}=0} = \begin{pmatrix} 1 & -(1-u_{ij}) \\ \frac{1-p_{ij}}{1-p_{ij}u_{ij}} & -\frac{1-p_{ij}-u_i+p_{ij}u_{ij}}{1-p_{ij}u_{ij}} \end{pmatrix} = \mathbf{B}_{ij}$$

$$if \quad k \in \Gamma(i) \setminus j \quad else \quad \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} \tag{17}$$

Now, we arrange all independent variables as a column vector $\mathbf{w} = (\mathbf{w}_{ij}, \cdots)^T$, where $\mathbf{w}_{ij} = (v_{ij}, y_{ij})^T$. The Jacobian matrix at zero input $\mathbf{w} = \boldsymbol{\vartheta}$ is

$$\mathbf{M} = \begin{pmatrix} \mathbf{M}_{klij} & \cdots \\ \vdots & \ddots \end{pmatrix} \tag{18}$$

Each block $\mathbf{M}_{klij}$ is

$$\mathbf{M}_{klij} = A_{ij} A_{kl} \delta_{il} (1 - \delta_{kj}) \mathbf{B}_{ij} \tag{19}$$

, where A_{ij} is the term in adjacency matrix between nodes i and j, and $\delta_{ij} = 1 \quad if \quad i = j \quad else \quad 0$.

The equation (19) is the same as (35) in [1] except that it is a matrix rather than scalar.

3.4. Calculation of Power Method and Definition of Collective Influence

As is shown in (16), the solution $\mathbf{w} = \boldsymbol{\vartheta}$ is stable only if the leading eigenvalue of $\mathbf{M}$ is less than 1. As is in [1], let us compute this leading eigenvalue by power method.

Start with one vector $|\mathbf{w}_0\rangle = ((\delta_{ij}, -\delta_{ij}), \cdots)^T$, the first order right vector is

$$|\mathbf{w}_1\rangle_{ij} = \sum_{kl} \mathbf{M}_{ijkl} |\mathbf{w}_0\rangle_{kl} = A_{ij} \sum_{jk, k \neq i} A_{jk} \mathbf{B}_{jk} \vec{v} \tag{20}$$

, where $\vec{v} = (1, -1)^T$.

The first order left vector is

$$\langle \mathbf{w}_1|_{ij} = \sum_{kl} \langle \mathbf{w}_0|_{kl} \mathbf{M}_{klij} = A_{ij}(d_i - 1)\vec{v}^T \mathbf{B}_{ij} \tag{21}$$

The first order inner product is

$$\langle \mathbf{w}_1|\mathbf{w}_1\rangle = \sum_{ij} A_{ij}(d_i - 1)\vec{v}^T \mathbf{B}_{ij} \left(\sum_{jk, k \neq i} A_{jk} \mathbf{B}_{jk} \right) \vec{v} \tag{22}$$

By generalized power method in [1](30)(54), the leading eigenvalue λ can be approximated by truncating the following equation at a certain number of iterations l.

$$\lambda = \lim_{l \to \infty} \left[\frac{\langle \mathbf{w}_l|\mathbf{w}_l\rangle}{\langle \mathbf{w}_0|\mathbf{w}_0\rangle} \right]^{\frac{1}{l}} \tag{23}$$

The structure of the inner product $\langle \mathbf{w}_0|\mathbf{M}^l|\mathbf{w}_0\rangle$ at the numerator is determined by the non-traceback property [1](13) of Jacobian matrix M, which means that M_{ijkl} is nonzero only if there are edges between ij, kl, and $j = k, i \neq l$. This corresponds to a flow of information from source node i via j to a different node l. The power of $\mathbf{M}$ take account of all such paths concatenated tail by head.

The flow cannot track back immediately after it travels through an edge. However, the restriction is lifted after it has accessed another node. Actually, a flow can return to a previously accessed node and trace back or pass through the same edge over and over again.

Fortunately, these complex interactions only contribute to a small fraction of the inner product. By [1], for a sparse network with local tree structure, all the contributions of paths containing cycles can be bounded to a $o(1)$ fraction of the sum for fixed l. When calculating the inner product, we only have to consider simple paths with length l.

Now we would derive a closed form expression of $\langle \mathbf{w}_0|\mathbf{M}^l|\mathbf{w}_0\rangle$ by induction. Suppose the l-th order right vector has the form

$$|\mathbf{w}_l\rangle_{ij} = A_{ij} \sum_{k: d(j,k) = l, i \notin p(j,k)} \left(\prod_{e \in p(j,k)} \mathbf{B}_e \right) \vec{v} \tag{24}$$

, where $d(i, j)$ is the distance between i and j, and $p(i, j)$ is the shortest path between them. (20) is a special case when $l = 1$.

Then $|\mathbf{w}_{l+1}\rangle = \mathbf{M}|\mathbf{w}_l\rangle$, and we neglect all paths that form cycles.

$$|\mathbf{w}_{l+1}\rangle_{ij} = \sum_{kl} \mathbf{M}_{ijkl} |\mathbf{w}_l\rangle_{kl} = A_{ij} \sum_{k:d(j,k)=l+1, i \notin p(j,k)} \left(\prod_{e \in p(j,k)} \mathbf{B}_e \right) \vec{v} \tag{25}$$

Similarly, we can get the expression for the l-th order left vector.

$$\langle \mathbf{w}_l|_{ij} = \sum_{k:d(k,i)=l-1, j \notin p(k,i)} (d_k - 1) \vec{v}^T \left(\prod_{e \in p(k,i)} \mathbf{B}_e \right) \mathbf{B}_{ij} \tag{26}$$

Therefore, when computing the inner product, all simple paths having $l-1$ edges before i and l edges after j are considered. We rename the start point of the concatenated path by i and the end point by j.

$$\langle \mathbf{w}_l|\mathbf{w}_l\rangle = \sum_i (d_i - 1) \sum_{j:d(i,j)=2l} \vec{v}^T \left(\prod_{e \in p(i,j)} \mathbf{B}_e \right) \vec{v} \tag{27}$$

, and

Actually, we may slightly alter the form of (23) so that the left vector and right vector are of different order and the power method will still converge to the leading eigenvalue. In this way, the length of the path in (27) can also be generalized to be odd numbers.

Then we would like to define the l-th order approximation of collective influence of each node in a similar manner as in [1](104)

$$CI_l(i) = (d_i - 1) \sum_{j:d(i,j)=l} \vec{v}^T \left(\prod_{e \in p(i,j)} \mathbf{B}_e \right) \vec{v} \tag{28}$$

To maximize λ with the least number of seeds, it is advisable to select nodes with greatest CI value.

By (23), as $\langle \mathbf{w}_0|\mathbf{w}_0\rangle = m$, where m is the number of edges in G. $\lambda > 1$ as long as

$$\langle \mathbf{w}_l|\mathbf{w}_l\rangle > m \tag{29}$$

3.5. Procedure of Seed Selection

Based on all the discussion above, now we would propose our algorithm for seed selection. This algorithm is a variant of collective influence in [1] tuned for independent cascade spreading models. We temporarily name it Collective Influence for Independent Cascade (CIIC).

Algorithm 1: Collective Influence algorithm for seed selection on Independent Cascade model. (CIIC)

Input: A graph $G(V,E)$ that models the social network, probability of infection spreading on each edge p_{ij}, truncating length l of CI computation, u_{ij} estimation accuracy ε, maximum number of seed nodes $K = n$ being number of nodes in network by default;

Output: A set of nodes S that should be enough to initiate an outbreak (or maximize influence within K nodes).

```
1 $S \leftarrow \varnothing$;
2 while $|S| < K$ and $\langle \mathbf{w}_l | \mathbf{w}_l \rangle < m$ do
3     Calculate the probability of infection $u_{ij}$, $u_i$ for each pair of nodes given seed set $S$ by (4) ;
4     Calculate the partial derivative matrix $\mathbf{B}_{ij}$ for $\forall i, j$ by (17) and then calculate the collective influence $CI_l(i)$ for $\forall i$ by (28) ;
5     Select the node $v \notin S$ with highest score of collective influence, and add it to $S$.
          $S \leftarrow S \cup \{\arg\max_{v \notin S} CI_l(v)\}$
6 end
7 return $S$.
```

3.6. Complexity Analysis of Our Algorithm

The time complexity of our algorithm is $O(Kn)$.

The outer cycle will repeat for K times. When l and ε are fixed and $p_{ij} < 1$, the computation of u_{ij} and $CI_l(i)$ is only based on a local region in the graph within a fixed radius. This region only contain $O(1)$ number of nodes when the degrees are bounded in G. Therefore, it takes $O(n)$ time to calculate u_{ij}, $\mathbf{B}_{ij}$, as well as $CI_l(i)$.

As only one node is selected as seed in each round. Only the variables that is affected by this addition need to be updated. When all variables are calculated locally, all these variables to update lie in a neighborhood of the newly selected seed. Running time could be further saved in this way.

The space complexity is $O(m)$. The variables that need to be stored are u_{ij}, u_i, S, $\mathbf{B}_{ij}$, and $CI_l(i)$. Storing all these variables only require memory space that is the same order as storing the graph G.

4. Validation by Experiments

We implement and validate our algorithm on *Python* based on a well-known libray *networkx.* We compare our algorithm with some common baseline algorithms on random graph and a practical network. The result shows that our algorithm can achieve better information spreading than those baseline algorithms. Section 4.1 will briefly introduce baseline algorithms and their implementation. In section 4.2 we introduce the optimization of CI and our algorihtm. Section 4.3 will show the result and analysis of the simulation experiment on random graph. The source code is available on *Github*[1].

4.1. Baseline Algorithms

High-Degree(HD) HD method ranks nodes directly according to the number of connections. In this experiment we define the influence of a node with its degree. We implement this algorithm as select the nodes with highest degree as seeds.

PageRank PageRank algorithms extends the idea in academic citation that the number of citations or backlinks give some approximation of a page's importance, by not counting links equally but normalizing by the number of links on a page. *networkx* provides us a function to obtain PageRank value for each node directly and we select those with highest PageRank vaule as seeds.

K-core The k-core is the largest subgraph where vertices have at least k interconnections. The k-knore may be obtained in the following way. Remove from a graph all vertices of degree less than k. Some of the rest vertices may remain with less than k edges. Then remove these vertices, and so on until no further removal is possible. The result, if it exists, is the k-core. In this project, we start from $k = 1$ to obtain a subgraph and increase k iteratively. If we need a set of seeds with size N, the last N nodes left in the graph through this process is what we need.

Collective Influence(CI) CI method inspects the collective influence of multiple spreaders. This method defines CI value of a node as follows:

$$CI_l(i) = (d_i - 1) \sum_{j \in Ball(i,l)} \Big(\prod_{k \in P_l(i,j)} \mu_k \rho_k \Big)(d_j - 1)$$

where d_i stands for the degree of node i. $Ball(i,l)$ consists of the nodes within a ball of radius l from node i(defined as the shortest path) and $P_l(k,j)$ is the shortest path of length

[1]`https://github.com/Huanghongru/EE447-Project`.

l from node i to j. CI method can be considered as a naive version of our algorithm since it doesn't involve the probability of successful transmission on edges. We select a node with the highest CI value as seed and remove it from the graph. Then we calculate CI value of each node in the new graph and repeat this procedure until we obtain enough seeds.

4.2. Optimization of CI and our algorithm

Theorectically, the computation complexity of CI is $O(N\log N)$. But in fact, it still takes much of time to pocess this algorithm. So we do some prunning when implementing CI algorithm. Since our algorithm is quite similar to CI except for the definition of collective influence of a node. This optimization method can also be applied to our algorithm.

The first idea is to find all the so-called ball-path of each node, which will save us a lot of time because after removal, we don't need to find the paths again.

The second idea is after removing a node with the highest CI value in the current graph, we only update those nodes that are affected by the removed node. These nodes can be the neighbor or in the ball-path of nthe removed node of the removed node.

4.3. Simulation Validation

In this section, we construct 4 different random graph as network on which we can test the performance of our algorithm. The random graph generated by *networkx* follows power law distribution, which can approximate the structure of pratical social network. The basic information of these 4 networks are shown in the table.

Table 3. Information of random graph

	Number of nodes	Number of edges	Average degree	Max degree
G1	1000	999	1.9989	32
G2	2000	1999	1.9990	45
G3	3000	2999	1.9993	53
G4	4000	3999	1.9995	68

On each network, we compare the performance of our algorithm with the baseline methods mentioned above. To do the simulation, we need to construct a virtual information spreading process. We firstly define state s_i for node i indicating whether it is infected. We set $s = 1$ for all seed nodes and $s = 0$ for all the other nodes in the graph as the initial state. The virtual information spreading process can mathematically describe as follows:

$$\forall j \in G - S, i \in S, s_j = 1 \text{ if } r > 1 - p_{i,j}$$

where r is a random variable follows uniform distribution over $[0,1]$. When compare the performance of different algorithms, this procedure may cause some problems. For a edge in the graph, the virtual information may pass successfully in HD method but fail in PageRank. To address this unfair circumstance, we construct a sampled graph G' using the following method before simulation.

$$\forall i, j \in G, p'_{i,j} = 1 \text{ if } r > 1 - p_{i,j} \text{ else } 0$$

Then we perform virtual information spreading on the sampled G' for fair comparison. The results are shown as follows:

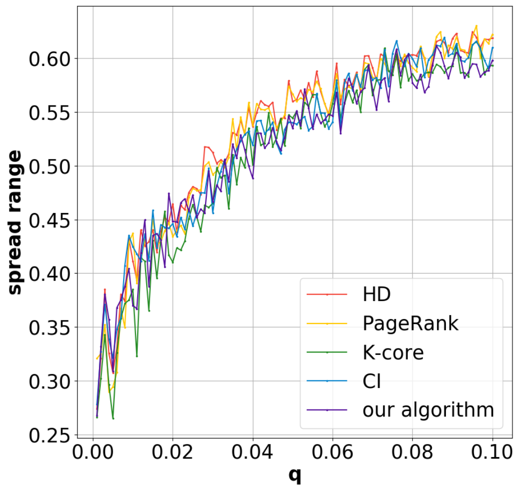

Figure 1. result of G1

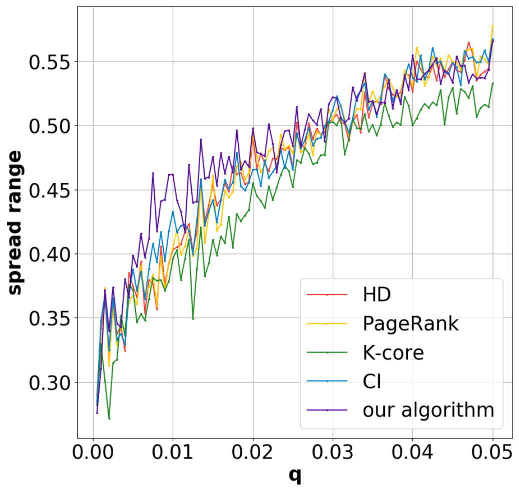

Figure 2. result of G2

As we can see, there is no perfomance difference in these algorithms in G1 because G1 is somewhat a small graph, which can not show the superiority of our algorithm. In the case of G2, G3 and G4, our algorithm performs better than other baseline algorithms. Take G3 as an example, in a small fraction of seed nodes, our algorithm perform much better than the k-core method. It also achieve higher spread range than the CI method, which doesn't consider the probability of successful transimission on the edge. In our simulation experiment, HD and PageRank have a similar performance but a little bit worse than our algorithm.

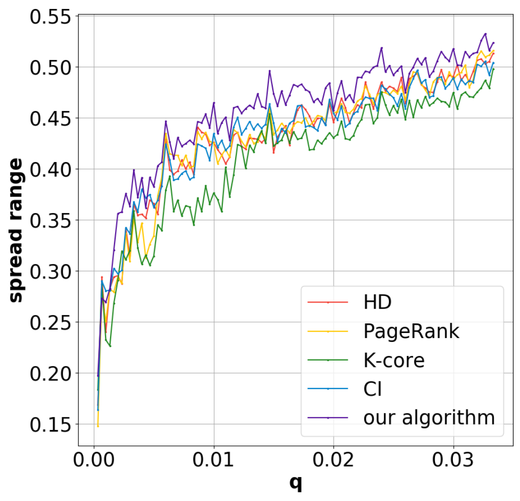

Figure 3. result of G3

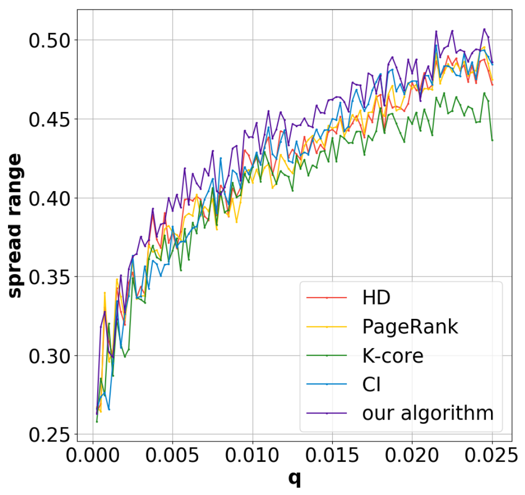

Figure 4. result of G4

5. Conclusion

We studied the influence maximization problem on independent cascade models, using the idea of collective influence to measure the influential powers of potential seed nodes.

We start from message passing formulas specific to independent cascade model, with 2 independent variables for each edge: one for the probability of belonging to the giant component of infected nodes, and the other for correlation of being infected and belonging to giant component.

The appearance of giant infected component has been related to the critical point where zero solution of message passing formulas become unstable. This is marked by a leading eigenvalue greater than 1. To infect as many nodes as possible, it is reasonable to select seeds to maximize the leading eigenvalue.

We compute the leading eigenvalue by power iteration. Due to the non-traceback property and neglecting short loops, we reach a collective influence formula that is similar in form as that in [1]. We then propose our algorithm for selecting the most influential nodes as seeds.

We conduct a simulation experiments on random graph that follows power law distribution. The simulation results show that with a pretty small fraction of seed nodes, our algorithm achieve a basic superiority over the baseline algorithms.

Currently, there are still many unfinished works in this issue. Calculating the number of seeds sufficient to initiate a giant component of infected nodes, and justifying consistency between maximizing the number of infected nodes and maximizing the probability of having a giant infected component are two such problems that remain unexplored.

Our algorithm also needs further improvements on reducing complexity. Calculating the probability of infection, as well as finding the local paths, is still very time consuming, limiting its performance and usage in large scale networks. For our algorithm to be of real value in influence maximization, higher levels of simplification is required, and more valid experimental data is needed to justify its performance.

6. Mission Division

Table 4. Mission Division

MENG Jingfan	theory derivation algorithm design
WANG Xuyao	our algorithm implementation experiment and analysis
HUANG Hongru	baseline algorithm implementation experiment and analysis

Note: we run simulation experiment on dozens of random graph with different characters(distribution, average degree, fractions of seeds and so on) to find out on which kind of graph our algorithm can achieve the best. This part of work is concluded as **experiment and analysis** simply, but it is quite large and tedious so it need two of our members to complete together.

References

[1] Hernán A. Makse Flaviano Morone. “Influence maximization in complex networks through optimal percolation”. In: *Nature* 524 (), p. 65. URL: `http://dx.doi.org/10.1038/nature14604` (cit. on pp. 1, 2, 4, 8–10, 15).

[2] Éva Tardos David Kempe Jon Kleinberg, ed. *Maximizing the Spread of Influence through a Social Network*. Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. 2003. DOI: `10.1145/956750.956769` (cit. on p. 2).