# Information Propagation In Social Networks

Divya Venkataramani
MIT College of Engineering,Pune
Email: v.divya110@gmail.com

Shubham Wadzirkar
MIT College of Engineering,Pune
Email: w.shubham2594@gmail.com

Jaikumar Ambekar
MIT College of Engineering
Email: mycustoms11@gmail.com

Aaishwarya Jadhav
MIT College of Engineering
Email: shwrjdhv48@gmail.com

***Abstract*—Influence maximization is a fundamental research problem in social networks. Influence maximization is the problem of finding a small subset of nodes (seed nodes) in a social network that could maximize the spread of influence. Viral marketing, one of its applications, is to get a small number of users to adopt a product, which subsequently triggers a large cascade of further adoptions by utilizing Word-of-Mouth effect in social networks. Time plays an important role in the influence spread from one user to another and the time needed for a user to influence another varies. In this paper we will be studying about various information propagation models by which ideas and influence propagate through a social network. We will also be studying that time constrained influence maximization is an NP hard problem and proof related to monotonicity and submodularity of time constrained influence maximization. Based on this, we will study the greedy algorithm. We will also study simulation based algorithm used to alleviate the computational complexity of greedy algorithm. We study the efficient influence maximization from two complementary directions. One is to improve the original greedy algorithm and its improvement to further reduce its running time, and the second is to propose new degree discount heuristics that improves influence spread. (a)our improved greedy algorithm achieves better running time (b) our degree discount heuristics achieve much better influence spread than classic degree and centrality-based heuristics, and when tuned for a specific influence cascade model, it achieves almost matching influence thread with the greedy algorithm. The Greedy algorithm for influence maximization suffers from two sources of computational deficiency: 1. The need to evaluate many candidate nodes before selecting a new seed in each round 2. The calculation of the influence spread of any seed set relies on Monte Carlo simulations. Hence we also study to tackle both these problems by devising efficient algorithms to compute influence spread and determine the best candidate for seed selection The fundamental insight behind the proposed algorithms is the linkage between influence spread determination and belief propagation on a directed acyclic graph (DAG).**

***Index Terms*—Influence maximization, Independent Cascade Model, Linear Threshold Model, Greedy Algorithm, DAG**

## I. Introduction

### A. Social Network

A social network is a structure which connects or ties together individuals or organizations based on their relationships. These relationships can be friendship, kinship, Financial dependency or relationship of belief[5]. A social network is used as a platform for dissemination of information amongst individuals who are connected by relationships.

### B. History Of Social Network

Social networking began in 1978 with the Bulletin Board System (or BBS.) The BBS was hosted on personal computers, in which the users required to dial through the modem of the host computer in order to exchange information over phone lines.

Later in the same year, Usenet was introduced by Jim Ellis and Tom Truscott. It allowed users to post news articles or posts, which were referred to as news. The first version of instant messaging was introduced in 1988 with Internet Relay Chat (IRC). IRC was Unix-based providing limited access. It was used for link and file sharing, and keeping in touch with one another.

Friendster was the pioneer of social networking. In the first three months, the social networking website acquired 3 million users. Friendster was the launching point for the widely popular MySpace, who replicated Friendster and launched after just 10 days of coding. In the following years, other social networking websites like Classmates.com, LinkedIn and Tribe.net were launched.

Facebook.com was launched in 2004 with the intention of connecting U.S. college students, starting with Harvard College. After gaining popularity, Facebook opened its registration to other students, and in 2008, Facebook surpassed MySpace as the leading social networking website. [4]

### C. Social Network Analysis

***Social network analysis (SNA)*** views social relationships in terms of network theory, which consists of nodes, representing individual actors within the network, and ties or edges which represent relationships between the individuals or organizations, such as friendship, relationship of belief, financial transactions. These networks are depicted in a social network diagram, in which nodes are represented as points and ties are represented as lines.

### D. Terminologies in terms of graph theory

A *graph* is a set of vertices(i.e nodes) V and a set of edges(or lines) E.

If an edge exists between (a,b) then we can say that nodes a and b are related to each other.

The edges can be unordered pairs of nodes or in a directed graph, ordered pairs of nodes where each edge has a direction(arc).

The world wide web can be contemplated as a very large graph where nodes could be individual sites or pages and edges could be the links between pages.
The importance of a site is determined by the number of sites that are linked to it weighted by the importance of the linked sites.
Importance propagates around the graph until it stabilizes, eventually we end up with probability that a random web surfer will be at a given page.

The degree of a vertex is the number of end edges at that vertex.
In a directed graph, the degree is usually divided into the in-degree and the out-degree(whose sum is the degree of the vertex in the underlying undirected graph) .[3]
**In-degree** of a vertex v can be defined as the number of edges with v as their terminal vertex.
The **out-degree** of a vertex v can be defined as the number of edges with v as their initial vertex. [3]

If we have a small network (graph) then we can analyze it visually by constructing its graph, however this is impossible for large networks.

Hence we use summary statistics and performance metrics to describe and compare networks and their graphs such as:

1) Diameter and mean path length
2) Centrality and nodal power
3) Degree distributions

The diameter of the graph is the largest distance between any 2 nodes.
If we let I(i,j) be the the length of the geodesic between nodes i and j then the diameter is the maximum I(i,j) over all possible node pairs.
The mean path length is the mean distance between all nodes in the graph.

Power is a fundamental property of social structures, related to centrality. Several techniques have been developed to study social power and we have 3 main measures of power or centrality:

1) **degree** - number of edges a given node has, its degree, normalised by total number of edges in graph.
2) **closeness** - average number of hops from a given node to all other nodes in the graph.
3) **betweeness** - the number of times that any node needs a given node to reach any other node by the shortest path.

### *E. Challenges of social network*

Suppose a small company develops a cool online application for an online social network and wants to market it through the network. It has a limited budget such that it can only select a small number of users in the network to use it. The company expects the users to start influencing their friends on the social network to use it and thus through the word-of-mouth effect a large audience in the social network would adopt the application.

The major issue is whom to choose as the initial users so that they influence the largest number of people in the network, i.e., the problem of finding influential individuals in a social network.

This problem is referred to as influence maximization and it is the main challenge for various organizations who have the ”word of mouth” effect.

Influence is propagated in the network according to a stochastic cascade model. There are three cascade models, namely the independent cascade model, weighted cascade model, and linear threshold model.

Consider a social network graph, with a specific influence cascade model, and a small number k, the influence maximization problem is to find k vertices in the graph (referred to as seeds) such that under the influence cascade model, the expected number of vertices influence spread by the k seeds is maximum.

Kempe et al. proved that the optimization problem is NP-hard, and presented a greedy approximation algorithm applicable to all three models, which guarantees that the influence spread. They also show through experiments that their greedy algorithm significantly outperforms the classic degree and centrality-based heuristics in influence spread.[5]

However, their algorithm has a serious drawback, which is its efficiency.

## II. Influence Maximization Problem

### *A. Influence Maximization*

Given a fixed budget a company can select only few users for spreading the influence about a specific product. The company then expects a cascade of influence spread by the word-of-mouth of the initial users.
The major problem is whom to target as initial users in order to get maximum spread of influence. This problem is considered as the influence maximization problem.

Influence maximization is of vital importance to companies and individuals in order to promote their products.
The solution to this problem is given by various algorithms, the most common of which is the ***Greedy Algorithm***.

### *B. The Greedy Algorithm*

## III. The Greedy Algorithm

Consider S as a subset of vertices selected to initiate the influence propagation, which we call the seed set. Let RanCas(S) represent the random process of influence cascade from the seed set S, of which the output is a random set of vertices influenced by S. Algorithms in this report take the graph G and a number k as input and generate a seed set S of cardinality k, with the intention that the expected number of vertices influenced by the seed set S, which we call influence spread, is as large as possible.

This algorithm describes the general greedy algorithm given a random process *RanCas()* . In each round i, the algorithm adds one vertex into the selected seed S such that this vertex together with current set S maximizes the influence spread(Line 10). Equivalently, this means that the vertex selected in round

**Algorithm 1** GeneralGreedy$(G, k)$

1: initialize $S = \emptyset$ and $R = 20000$
2: **for** $i = 1$ to $k$ **do**
3: **for** each vertex $v \in V \setminus S$ **do**
4: $s_v = 0$.
5: **for** $i = 1$ to $R$ **do**
6: $s_v \mathrel{+}= |RanCas(S \cup \{v\})|$
7: **end for**
8: $s_v = s_v / R$
9: **end for**
10: $S = S \cup \{\arg\max_{v \in V \setminus S}\{s_v\}\}$
11: **end for**
12: output $S$.

Fig. 1. General Greedy algorithm

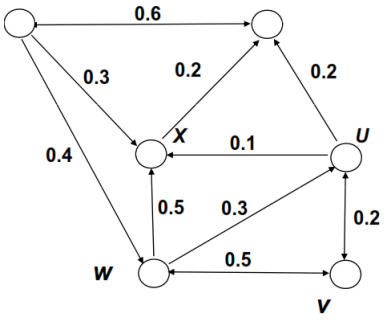


Fig. 2. Graph for Propagation Model

i is the one that maximizes the incremental influence spread in this round.

To do so, for each vertex v $\notin$ S, the influence, the influence spread of S $\cup\{v\}$ is estimated with R repeated simulations of RanCas(S $\cup\{v\}$) (Lines 3-9). Each calculation of RanCas(S) takes O(m) time, and thus Algorithm 1 takes O(knRm) time to complete.

## IV. Information Propagation Model

For maximizing the spread of Influence through a social network two propagation models are implemented.The two propagation models are as follows:

1) Independent Cascade Model[1]
2) Linear threshold Model[1]

### A. Independent Cascade Model

Consider a node v in a network,When node v becomes active it has only one attempt for activating the currently inactive neighbour node w. The probability for successful activation of neighbouring node is denoted $p_{vw}$.[1]

Independent cascade Model (ICM) is a type of stochastic information diffusion model.Stochastic information model in which the information flows over the network in cascades.Independent cascade model has two types of nodes

1) Active nodes
2) Inactive nodes

**Active nodes**: These nodes are already influenced by information in diffusion

**Inactive nodes**: These nodes are influenced by the the active nodes in diffusion process.

ICM process runs in discrete steps.At the initialization stage in ICM process few nodes are provided with information these nodes are considered as **SEED NODES**.In each discrete step one of the active node influences its neighboring inactive nodes. In cascade model the same active node never gets chance to influence the same inactive neighboring node. The successful activation from active node to an inactive node depends propagation probability.It is a probability by which on can influence the other node.It is relation dependent each edge has a different value.

### B. Linear Threshold Model

A node v is influence by each neighbor w according to a weight $b_{v,w}$ such that

$$\sum b_{v,w} \leq 1$$

Each node has v has threshold $\theta_v$ which is selected randomly from random [0,1] A node v becomes active if

$$\sum b_{v,w} \geq \theta_v$$

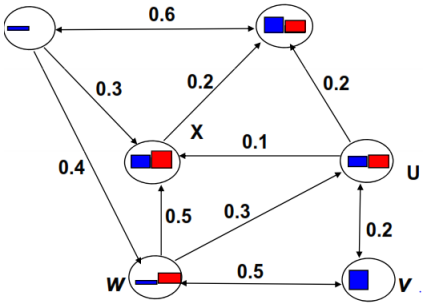


Fig. 3. Linear Threshold Models

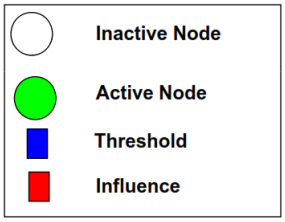


Fig. 4. Notations In Linear Threshold Model

## V. Problem Definition and The Greedy Algorithm

A social network is depicted as an undirected graph G = (V,E), with vertices in V depicting the individuals in the

network and edges in E depicting the relationship between individuals.

For example, consider coauthorship graphs where vertices are authors of papers then two vertices have an edge if the two corresponding authors have coauthored a paper.

We use n to represent the number of vertices and m to represent the number of edges throughout the paper. For convenience, Table 2.1 lists important variables used throughout the report.[2]

Table 2.1: **Important variables used in the report**

| Variables | Description |
|---|---|
| n | number of vertices in G |
| m | number of edges in G |
| k | number of seeds to be selected |
| R | number of rounds of simulations in Algorithms 1,2 and 3 |
| T | number of iterations in the Cohen's algorithm used in algorithm 3 |
| p | propagation probability in the IC model |
| $d_v$ | degree of vertex v in G |
| $t_v$ | number of neighbours of vertex v already selected in seeds |

## VI. INFLUENCE PROPAGATION MODEL AND THE TIME CONSTRAINED INFLUENCE MAXIMIZATION PROBLEM

### A. Conventional Influence Maximization Problem

In conventional influence maximization problem the aim is to select K nodes so that expectedd number of nodes influenced by K nodes will be maximized.

*1) Definition of influence maximization problem:* Given a social network $G = (V, E)$ a positive integer $K < |V|$, activating probability $P_{uv} \in (0, 1]$ for each $(u, v) \in E$, we have to find a seed set S $\subset$ V K nodes, such that the expected number of nodes influenced by S, $\sigma$ T(S), is maximized.

*2) Solution to conventional influence maximization problem:* One of the popular models describing how influence spreads in social networks is the Independent Cascade (IC) Model[1]. It is widely used by the already existing influence maximization algorithms.

But, some delay relating to influence propagation exists in a real-word social network, which is not captured by the IC model. We proceed to present the Latency Aware Independent Cascade (LAIC) model [1]. It encodes the influence propagation latency information into the IC model.

### B. Latency Aware Independent Cascade (LAIC) Model

In LAIC model, when a node u is first activated at step t, it activates its currently inactive neighbour, v, in step t+$\delta$t with a probability $P_{uv}P_u^{lat}(\delta)$t, where $\delta$t is the influencing delay and is randomly drawn from the delay distribution $P_u^{lat}$.

Note that a node can be activated at most once only. If a node has multiple neighbors influencing it, it is activated at the earliest activation time while the rest activations are discarded. The process of influence propagation terminates at step t, iff there is no node activated after t.

### C. Time Constrained Influence Maximization Problem

Based on the LAIC model, we now present the time constrained influence maximization problem.

*1) Definition:* Given a social network G=V,E, time bound T, positive integer $K < |V|$, activating probability $P_{uv} \in (0, 1]$ for each (u,v) $\in$ E, and latency distribution $P_u^{lat}$ for each u $\in$ V, find a seed set S $\subset$ V of K nodes, such that the expected of nodes influenced by S within T time, $\sigma$T(S), is maximized under the LAIC model.

***Theorem 1:****The time constrained influence maximization problem is* NP-Hard

***Proof:*** The traditional influence maximization problem is known to be *NP-Complete*. It is the corresponding time constrained problem with unlimited time bound, we argue that the traditional influence maximization problem is special case of time constrained influence maximization problem, which thus is *NP-Complete*.

*2) Monotonicity and submodularity of time constrained influence function:* Here we show the Monotonicity and Submodularity of the time constrained Influence function $\sigma$T(S) in the LAIC model, which leads us to a natural hill climbing greedy framework.

Let $\sigma_T$(S) be the expected number of nodes influenced within time T units. Now, by replacing $\sigma$(S) with $\sigma_T$(S), th greedy algorithm[1] is adapted to approximately solve the ttime constrained influence maximization problem, which is given in Algorithm 1. [1]

### D. The Greedy Algorithm

The greedy algorithm one after the other, keeps on adding the node incurring the largest marginal influence increase to seed set *S*, until [S]=K.

**Algorithm 1:** Greedy Algorithm Framework

**Input**: $\mathcal{G}$, $T$, $K$, $\mathcal{P}_{uv}$ and $\mathcal{P}_u^{lat}$
**Output**: $S$
initialize $S = \emptyset$
**for** $i \leftarrow 1$ **to** $K$ **do**
  $u \leftarrow \arg\max_v \sigma_T(S \cup \{v\}) - \sigma_T(S)$
  $S \leftarrow S \cup \{u\}$
**end**

The time complexity of Algorithm 1 is O($K_n$T($\sigma_T$(S))), where *n* is the number of nodes in *G* and *T($\sigma_T$(S))* the running time for calculating $\sigma_T$(S$\cup$v). As Theorem 2 shows the influence function $\sigma_T$(S) is monotonous and submodular, and thus greedy algorithm approximates the optimal solution with a lower bound ratio of 1-1/e.[3]

***Theorem 2:*** *With the LAIC model, the influence function $\sigma_T$(S) is monotonous and submodular.*

The main difficulty in applying the greedy algorithm lies in calculating the expected influence spread for a given fet of

seeds (Line 3 of Algorithm 1), whose special case has been shown to be #*P-hard*. Further we present a simulation based algorithm [2].

**Algorithm 2:** $\sigma_T(S)$ based on Simulation

**Input**: $\mathcal{G}$, $T$, $S$, $\mathcal{P}_{uv}$ and $\mathcal{P}_u^{lat}$
**Output**: $\sigma_T(S)$

1 $v.status \leftarrow$ ***inactive***, $v.actTime \leftarrow +\infty$ for $v \in \mathcal{V} \setminus S$
2 $v.status \leftarrow$ ***active***, $v.actTime \leftarrow 0$ for $v \in S$
3 $A_0 \leftarrow S$
4 $t \leftarrow 1$
5 **do**
6  **for** $u \in A_{t-1}$ **do**
7  **for** $(u, v) \in \mathcal{E}$ *and* $v.status \neq$ ***active*** **do**
8  draw $flag$ from $Bernoulli(\mathcal{P}_{uv})$
9  **if** $flag = 1$ **then**
10  draw $\delta_t$ from $\mathcal{P}_u^{lat}$
11  **if** $v.status =$ ***inactive*** **then**
12  **if** $t + \delta_t \leq T$ **then**
13  $v.status \leftarrow$ ***latent active***
14  $v.actTime \leftarrow t + \delta_t$
15  **end**
16  **end**
17  **else if** $t + \delta_t < v.actTime$ **then**
18  $v.actTime \leftarrow t + \delta_t$
19  **end**
20  **end**
21  **end**
22  **end**
23  $A_t \leftarrow \{u | u.actTime = t \cap u.status =$ ***latent active***$\}$
24  $u.status \leftarrow$ ***active*** for $u \in A_{t-1}$
25  $t \leftarrow t + 1$
26 **while** $|\{u | u.status =$ ***latent active***$\}| \neq 0$ *or* $A_t \neq \emptyset$;
27 return $\sum_{j=0}^{t} |A_j|$

*1) Simulation based algorithm for $\sigma_T(S)$:* We present Algorithm 2 to simulate the time constrained influence spreading process based on time steps. Algorithm 2 is different from simulation algorithm for conventional influence maximization problem, which is based on Breadth-First Search and does not consider time factor.

In Algorithm 2, we present the simulation of the influence propagation process starting from *S*. In the beginning, all the nodes in *S* are set as *active*, and all the other nodes are set as *inactive*(Line 1-2 of Algorithm 2).

Node set activated at *t* time is denoted by $A_t$. All the nodes in *S* are being activated at time 0(Line 3).

At the time step *t*> 0, each node $u \in A_{t-1}$ intends to activate each of its *latent active* or *inactive* outgoing neighbors $v \in N_{out}$(u) which have a probability $P_{uv}$.

If *u* successfully activates *v*(Line 9-20), activating latency $\delta_t$ ($\delta_t$=0,1,2...) is drawn from the discrete distribution $P_u^{lat}$ associated with node *u*. If *v* is in *inactive* state and *t*+$\delta_t \leq T$, *v switches to latent active state with activating time t*+ $\delta_t$, which specifies when *v* will switch from *latent active* to *active*. If *v* is already in *latent active* state, *v* updates its activating time with the minimum of *t*+ $\delta_t$ and its current activating time. All *latent active* nodes with activating time *t* automatically switch to *active* state at time step *t* (Lines 23-24). The process terminates if and only if there is no more *latent active* nodes and newly activated nodes. When the process terminates, the number of activated nodes is returned (Line 27).

## VII. Improvised Greedy Algorithm

Consider S as a subset of vertices selected to initiate the influence propagation, which we call the seed set. Let RanCas(S) represent the random process of influence cascade from the seed set S, of which the output is a random set of vertices influenced by S. Algorithms in this report take the graph G and a number k as input and generate a seed set S of cardinality k, with the intention that the expected number of vertices influenced by the seed set S, which we call influence spread, is as large as possible.

**Algorithm 1** GeneralGreedy$(G, k)$

1: initialize $S = \emptyset$ and $R = 20000$
2: **for** $i = 1$ to $k$ **do**
3:  **for** each vertex $v \in V \setminus S$ **do**
4:  $s_v = 0$.
5:  **for** $i = 1$ to $R$ **do**
6:  $s_v \mathrel{+}= |RanCas(S \cup \{v\})|$
7:  **end for**
8:  $s_v = s_v / R$
9:  **end for**
10:  $S = S \cup \{\arg\max_{v \in V \setminus S}\{s_v\}\}$
11: **end for**
12: output $S$.

Fig. 5. General Greedy algorithm

This algorithm describes the general greedy algorithm given a random process *RanCas()* . In each round i, the algorithm adds one vertex into the selected seed S such that this vertex together with current set S maximizes the influence spread(Line 10). Equivalently, this means that the vertex selected in round i is the one that maximizes the incremental influence spread in this round.

To do so, for each vertex v $\notin$ S, the influence, the influence spread of S $\cup\{v\}$ is estimated with R repeated simulations of RanCas(S $\cup\{v\}$) (Lines 3-9). Each calculation of RanCas(S) takes O(m) time, and thus Algorithm 1 takes O(knRm) time to complete. [1]

Time complexity of all algorithms is summarized in Table 2.2 for convenience of comparison. In [6], Leskovec et al. presents

TABLE I
**Time complexity of algorithms**

| Algorithms | Time complexity |
|---|---|
| Algorithm 1: General Greedy | O(knRm) |
| Algorithm 2: NewGreedyIC | O(kRm) |
| Algorithm 3: NewGreedyWC | O(kRTm) |

a CELF optimization to the original greedy algorithm based on the the submodularity of the influence maximization objective. The submodularity property is that when adding a vertex v into a seed set S, the incremental influence spread as the result of adding v is larger if S is smaller. CELF optimization

utilizes submodularity such that in each round the incremental influence spread of a large number of nodes do not need to be re-evaluated because their values in the previous round are already less than that of some other node evaluated in the current round.

CELF optimization is 700 times faster than Greedy Algorithm

**Algorithm 2** NewGreedyIC$(G, k)$

```
initialize S = ∅ and R = 20000
for i = 1 to k do
    set s_v = 0 for all v ∈ V \ S
    for i = 1 to R do
        compute G' by removing each edge from G with probability 1 − p
        compute R_G'(S)
        compute |R_G'({v})| for all v ∈ V
        for each vertex v ∈ V \ S do
            if v ∉ R_G'(S) then
                s_v += |R_G'({v})|
            end if
        end for
    end for
    set s_v = s_v/R for all v ∈ V \ S
    S = S ∪ {arg max_{v∈V\S} {s_v}}
end for
output S
```

*2) Improvement for the independent cascade model[2]:* In the IC model, RanCas(S) works as follows: Let $A_i$ be the set of vertices that are activated in the i-th round , and $A_0$ = S. For any $\overline{uv} \in$ E such that u $\in A_i$ and v is not yet activated, v is activated by u in the(i+1)th round with an independent probability p, which we call the *propagation probability*.
In the other words, if there are $l$ neighbours of v that are in $A_i$, v $\in A_{i+1}$ with probability $1 - (1 - p)^l$. This process is repeated until $A_{i+1}$ is empty.

Notice that in the random process RanCas(S), each edge $\overline{uv}$ is determined once, either from u to v or from v to u, on whether the influence is propagated through this edge. Moreover, the probability on either direction is the same propagation probability p. Therefore, we may determine first whether $\overline{uv}$ is selected for propagation or not, and remove all edges not for propagation from G to obtain a new graph G'. With this treatment, the random set RanCas(S) is simply the set of vertices reachable from S in G'. Let $R_{G'}(S)$ denote the set of vertices reachable from S in graph G'. The advantage of generating G' first is that, with a linear scan of the graph G' (by either DFS or BFS), we not only obtains $R_{G'}(S)$, but we can also obtain the size of $R_{G'}(v)$ for all vertices v $\in$V . Then, for every v $\in$ V \S, the additional number $s_v$ of vertices in G' that are influenced by selecting v into the seed set S is either $|R'_G(\{v\})|$ if v $\notin R_{G'}(S)$ or 0 if v $\in R_{G'}(S)$.

Therefore, by randomly generating G' for R times, and each time computing $s_v$ as stated above for all v $\in$ V \ S by a linear scan of graph G', we can select the next best candidate vertex v with the best average $s_v$. Algorithm 2 gives the details of the above improved algorithm. Since computing $R_{G'}(S)$ and $R_{G'}(\{v\})$ for all vertices v $\in$ V takes O(m) time, the running time of the algorithm is O(kRm) where R is the number of simulations. Therefore, our improvement in Algorithm 2 provides O(n) speedup to the original greedy algorithm.[1]

In NewGreedyIC, each random graph is used to estimate the influence spread of all vertices, which may cause correlations among influence spread estimates.
However, we believe that these correlations are insignificant, because

1) they do not affect the estimate of each individual vertex,
2) correlations are mainly generated due to vertices coexisting in the same connected component of some random graphs, which are small comparing to the graph size, and
3) the estimate is taken as an average from a large number of random graphs (e.g. R = 20000), and thus for every pair of vertices they coexist only in a small portion of random graphs sampled.

[1]

Comparing our NewGreedyIC algorithm with the CELF optimization, there is a tradeoff in running time. In the CELF optimization, its first round is as slow as the original algorithm. However, starting from the second round, each round may only need to explore a small number of vertices and the exploration of each vertex is typically fast since RanCas(S) usually stops after exploring a small portion of the graph.
In contrast, in every round of our NewGreedyIC algorithm, we need to traverse the entire graph R times to generate R random graphs G'. To combine the merits of both improvements, we further consider the MixGreedyIC algorithm, in which in the first round we use NewGreedyIC to select the first seed and compute influence spread estimates for all vertices, and then in later rounds we use the CELF optimization to select remaining seeds.[1]

### A. Improvement in weighted cascade(WC) model

Let $d_v$ be the degree of v in graph G, and let $\overline{uv}$ be an edge in G.

In the weighted cascade (WC) model, if u is activated in round i, then with probability 1 / $d_v$, v is activated by u in round i+1. Similar to the IC model, each neighbor can activate v independently.
Therefore, if a not-yet-activated vertex v has l neighbors activated during the i-th round, the probability that v is activated in round i + 1 is $1 - (1 - 1/d_v)^l$.[1]

Let $d_v$ be the degree of v in graph G, and let $\overline{uv}$ be an edge in G. In the weighted cascade (WC) model, if u is activated in round i, then with probability $1/d_v$, v is activated by u in round i+1. Similar to the IC model, each neighbor can activate v independently. Therefore, if a not-yet-activated vertex v has l neighbors activated during the i-th round, the probability that v is activated in round i + 1 is $1 - (1 - 1/d_v)^l$.

The major difference between RanCas(S) in the WC model and the IC model is that the probability of u activating v is usually not the same as the probability of v activating u. Because of this, we build a directed graph $\hat{G}$= (V, $\hat{E}$), in which

**Algorithm 3** NewGreedyWC$(G, k)$

initialize $S = \emptyset$, $R = 20000$, $T = 5$.
**for** $i = 0$ to $k$ **do**
  initialize $s_v = 0$ for all vertices.
  **for** $j = 1$ to $R$ **do**
    obtain $G' = RanWC(G)$
    compute DAG $G'^*$ and weights $w(v^*)$ for all $v^* \in V^*$
    **for** $\ell = 1$ to $T$ **do**
      for each $v^* \in V^*$, $s^{\ell}_{v^*} = 0$
      for each $v^* \in V^*$, generate random value $X^{\ell}_{v^*}$ from the exponential distribution with mean $1/w(v^*)$
      for each $v^* \in V^*$, compute $Y^{\ell}_{v^*} = \min_{u^* \in R_{G'^*}(S^* \cup \{v^*\})} X^{\ell}_{u^*}$
      for each $v^* \in V^*$, $s^{\ell}_{v^*} += Y^{\ell}_{v^*}$
    **end for**
    for each $v \in V \setminus S$, $s_v += (T-1)/s^{\ell}_{v^*}$
  **end for**
  set $s_v = s_v/R$ for all $v \in V \setminus S$
  $S = S \cup \{\arg\max_{v \in V \setminus S}\{s_v\}\}$
**end for**
output $S$

Fig. 6. New Greedy WC

each edge $\overline{uv} \in$ E is replaced by two directed edges $\overrightarrow{uv}$ and $\overrightarrow{vu}$. We still use $d_v$ to denote the degree of v in the original graph.

The major difference between RanCas(S) in the WC model and the IC model is that the probability of u activating v is usually not the same as the probability of v activating u. Because of this, we build a directed graph $\hat{G}$= (V,$\hat{E}$), in which each edge $\overline{uv} \in$ E is replaced by two directed edges $\overrightarrow{uv} and \overrightarrow{vu}$. We still use $d_v$ to denote the degree of v in the original graph.

Using the same idea from the IC model, in each round of the greedy algorithm when selecting a new vertex to be added into the existing seed set S, we generate R random directed graphs G' = RanWC($\hat{G}$). For each vertex v and each graph G', we want to compute $|R_{G'}$ (S $\cup\{v\})|$, and then average among all G' to obtain the influence spread of S $\cup\{v\}$ and select v that maximizes this value. However, the algorithm differs from the IC model in its time complexity of computing $|R_{G'}(S \cup \{v\})|$ for all vertices v. In the IC model, it takes O(m) time total since G' is an undirected graph. In the WC model, G' is a directed graph, making the algorithm non-trivial. A straightforward implementation using BFS from all vertices take O(mn) time, or using fast binary matrix multiplication takes O($n^{2:38}$) [8], which is not as good as O(mn) for sparse graphs such as social network graphs. To solve this problem, we adapt the randomized algorithm of Cohen [7] for estimating the number of all reachable vertices from every vertex.

We now explain Cohen's algorithm in brief. Given a directed graph G', the first step is to traverse the graph once, compute all strongly connected components of G', and collapse each strongly connected component into one vertex with the weight being the size of the strongly connected component.
Let G'* denote the resulting directed acyclic graph (DAG), and let V* denote the vertex set of G'*. For any v∈V , let v* denote the corresponding (collapsed) vertex in V*
Let S* = v* ∈ V — v ∈ S. Let w(v*) denote the weight of v* in G'*.
Thus, in O(m) time we obtain a new directed acyclic graph (DAG) G'* such that every vertex v* has a weight w(v*). For S ⊆ V
, Let w(S*) =$\Sigma_{v* \in S*}$ w(v*). One important property of G'* with weights w() is that w($R_{G'*}(S* \cup \{v*\})$ = $|R_{G'}$ (S $\cup\{v\})|$. Cohens algorithm estimates $|R_{G'}$ (S $\cup\{v\})|$ in T iterations. In the i-th iteration, on every vertex v* ∈ V* we take a random sample $X^i_{v*}$ according to an exponential distribution with the probability density function w(v*) $e^{w(v*)x}$, x ≥ 0 (one way to obtain the sample is to sample z uniformly from [0, 1] and output -(ln z)=w(v*)). Then we compute

THEOREM 1 (COHEN [1]). *For every vertex $v$ in graph $G'$,*
*for* $0 < \epsilon < 1$ , $\text{Prob}[|\hat{W}_v - W_v| \geq \epsilon W_v] = exp(-\Omega(\epsilon^2 T))$,
*for* $\epsilon \geq 1$ , $\text{Prob}[|\hat{W}_v - W_v| \geq \epsilon W_v] = exp(-\Omega(\epsilon T))$. [1]

Fig. 7. Cohen's algorithm

We incorporate Cohens algorithm into our greedy algorithm such that for each of the R generated graphs G' in each of the k rounds, Cohens algorithm is run with T iterations to estimate $|R_{G'}(S \cup \{v\})|$ for all vertices v. The overall complexity is O(kRTm). Comparing with the complexity O(kRnm) of the original greedy algorithm, it is more efficient if T = o(n).
Indeed, in our experiments we show that a fairly small value of T = 5 already achieves very good estimates of the influence spread. The reason is that in the outer loop we will take R = 20000 of these estimates and average them, and thus the inaccuracy of each estimate due to small T is canceled out. The details of the algorithm is given in Algorithm 3. Similar to the case in the IC model, we also consider the Mixed- GreedyWC algorithm where the first round uses NewGreedyWC algorithm and the remaining rounds use the CELF optimization.
Since each round of NewGreedyWC needs O(TR) times of graph traversals, the mixed strategy makes more sense. Indeed, our experimental results show that it is the best in terms of the running time.[1]

## VIII. CONCLUSION

Big data analytics has the potential to transform the way healthcare providers use sophisticated technologies to gain insight from their clinical and other data repositories and make informed decisions. In the future we will see the rapid, widespread implementation and use of big data analytics across the healthcare organization and the healthcare industry. To that end, the several challenges highlighted above, must be addressed. As big data analytics becomes more mainstream, issues such as guaranteeing privacy, safeguarding security, establishing standards and governance, and continually improving the tools and technologies will garner attention. Big data analytics and applications in healthcare are at a nascent stage of development, but rapid advances in platforms and tools can accelerate their maturing process.

The Greedy algorithm runs the ***Monte Carlo Simulation*** of the influence cascade model multiple times until an accurate estimate of the influence spread is obtained. Hence, it takes days to calculate a small seed set in a moderately large network

However, their algorithm has a serious drawback, which is its efficiency.

This efficiency in algorithm is explained in this seminar. In this work,we first establish the link between influence spread computation and belief propagation on a Bayesian network(modeled as a directed acyclic graph-DAG) where marginal conditional dependancy corresponds to the influence probabilities.

## IX. Bayesian Network

A Bayesian network, belief network,l or a probabilistic directed acyclic graphical model is a probabilistic graphical model (a type of statistical model) that represents a set of random variables and their conditional dependencies via a directed acyclic graph(DAG)

For example,a Bayesian network could represent the probabilistic relationships between diseases and symptoms. Given symptoms, the network can be used to compute the probabilities of the presence of various diseases.

Formally, Bayesian networks are DAGs whose nodes represent random variables in the Bayesian sense: they may be observable quantities, latent variables, unknown parameters or hypotheses. Edges represent conditional dependencies; nodes that are not connected represent variables that are conditionally independent of each other.

Each node is associated with a probability function that takes, as input, a particular set of values for the node's parent variables, and gives (as output) the probability (or probability distribution, if applicable) of the variable represented by the node. For example, if parent nodes represent Boolean variables then the probability function could be represented by a table of entries, one entry for each of the possible combinations of its parents being true or false. Similar ideas may be applied to undirected, and possibly cyclic, graphs; such are called Markov networks.

### A. Belief Propogation

Belief propagation, also known as sum-product message passing is a message passing algorithm for performing inference on graphical models, such as Bayesian networks and Markov random fields. It calculates the marginal distribution for each unobserved node, conditional on any observed nodes[2].

Belief propagation is commonly used in artificial intelligence and information theory and has demonstrated empirical success in numerous applications including low-density parity-check codes, turbo codes and free energy approximation If X=Xi is a set of discrete random variables with a joint mass function p, the marginal distribution of a single Xi is simply the summation of p over all other variables:

$$p_{X_i}(x_i) = \sum_{\mathbf{x}' : x'_i = x_i} p(\mathbf{x}').$$

Fig. 8. Equation for BP

## X. Influence Spread on a DAG

### A. Problem Structure

Consider a directed graph G = (V,E) with V = n vertices and E = m edges. For every edge (u,v) $\in$ E, p(u,v) denotes the probability of influence being propagated on the edge. We adopt the Independent Cascade model.

Given a seed set S $\subseteq$ V , the IC model works as follows. Let St $\subseteq$ V be the set of node (newly) activated at time t, with So = S and St $\cap$ St-1 = $\emptyset$. At round t+1, every node u $\in$ St tries to activate its neighbors in v $\in V \bigcup_{0 \leq i \leq t} S_i$ independently with probability p(u,v). The influence spread of S, denoted by $\sigma(S)$, is the expected number of activated nodes given seed set S. Two important properties of the $\sigma()$ function are:

- $\sigma()$ is sub-modular.
- $\sigma$(S) is monotone, i.e. $\sigma(S) \leq \sigma(T)$ for all set S $\leq$ T.For any given spread function $\sigma()$ that is both sub-modular and monotone, the problem of finding a set S of size k that maximizes $\sigma$(S) can be approximated by a simple greedy approach.

### B. Calculating $\sigma()$ via Belief Propogation

For singly-connected DAGs, where between any two vertices there is only one simple path, the belief propagation (BP) algorithm [3] computes the exact solution with O(n) complexity. For multi-connected DAGs, where multiple simple paths may exist between two vertices, belief propagation and many of its variants [2] have been shown to work well in general.

BP algorithms take as input a factor graph or a Bayesian Network. For each factor in the graph or a Bayesian node, a conditional probability table(CPT) is constructed. For a node v with the parent set Par(v) = (par1,par2,...,park) its CPT consists of one column for each state and one row for each set of states its parents may assume. In influence spread, each state has two states: active (1) and inactive(0). Thus the number of rows in a CPT is $2^k$. An illustrative example of a factor graph and one of its CPT's is given

Once the factor graphs and the associated CPT's of every factor graph are calculated,we can apply a suitable BP algorithm to calculate the active probability of each node in the DAG.$\sigma()$ can then be determined

*1) Computation Complexity:* The complexity of $\sigma()$ calculation is dominated by the execution of the BP algorithm. A variety of BP algorithms exist.The Loopy Belief Propagation (LBP) algorithm which was shown to perform well for various problems[3,4]. LBPs complexity to estimate the active probability of a node v is O(Md), where

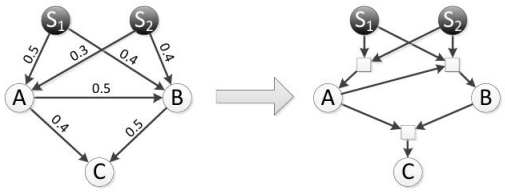


Fig. 9. Converting DAG to Factor Graph

| | | States of $C$ | |
|---|---|---|---|
| $A$ | $B$ | 0 | 1 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0.5 | 0.5 |
| 1 | 0 | 0.6 | 0.4 |
| 1 | 1 | 0.3 | 0.7 |

Fig. 10. CPT of C with two parents A and B

1) M is the number of possible labels (states) for each variable (M = 2)
2) d is the number of neighbors of v

### C. Single Pass Belief Propagation Heuristic

Calculating $\sigma()$ with LBP produces highly accurate results, but the computation time remains to be high when the graph is multi-connected. The main complexity arises from the fact that the activation of parents of a node may be correlated in a multi-connected graph. Thus, in computing the activation probability of the node, one needs to account for the joint distribution of its parent nodes.

Next, we propose a single pass belief propagation (SPBP) algorithm that ignores such correlation in determining $\sigma()$. Note that the heuristic is exact when the graph is singly-connected. Let D() be the input DAG. Consider a node v $\in$ D(). Given the activation probabilities of its parents Par(v), we approximate p(v) as .

$$p(v) = 1 - \prod_{u \in Par(v)} (1 - p(u)p(u,v)).$$

Fig. 11. Equation for approximating p(v)

The algorithm is summarized in Algorithm 1. It starts with the seed nodes and proceeds with the topological sorting order. Clearly, the algorithm has a complexity of O(d.n), where d is the maximum in-degree.

## XI. Building a DAG

In general, real social networks are not DAGs (with the exception of advisor-advisee and parent-child relationship, for instance, which exhibit a natural hierarchy). To apply the BP algorithm in computing influence spread, one needs to selectively prune edges and reduce the graph to a DAG.

**Algorithm 1:** Single-Pass Belief Propagation (SPBP)

```
input : D(S)
σ(S) = 0;
foreach v ∈ D(S) do
    if v ∈ S then
        p(v) = 1
    else
        p(v) = 1 − ∏_{u∈Par(v)} (1 − p(u)p(u, v))
    σ(S) = σ(S) + p(v)
output: σ(S)
```

The challenge is to find a DAG that approximates well the original graph in influence spread. In this section, we introduce two DAG construction algorithms, both retaining important edges where influences are likely to travel.

## XII. Influence path definitions

One important observation in [1] is that the influence of a seed node diminishes quickly along a path away from the seed node. In other words, the **perimeter** of influence or the influence region of a seed node is in fact very small. One way to characterize the influence region of a node v is through the union of maximum influence paths defined next.

*1) Path Propogation Probability:* For a given path P(u,v) = u1,u2,...,ul of length l from a vertex u to v, are intermediate vertices, define the propagation probability of the path, p(P), as:

$$p(P(u,v)) = \prod_{i=1}^{l-1} p(u_1, u_{i+1}).$$

Fig. 12. Equation for Path Propogation Probability

*2) Maximum Influence Path:* Denote by P(G,u,v) the set of all paths from u to v in G. The maximum influence path MIP(G,u,v) from u to v is defined as:

$$MIP(\mathcal{G}, u, v) = \arg\max_{P} \{p(P) | P \in \mathcal{P}(\mathcal{G}, u, v)\}.$$

Fig. 13. Equation for MIP

*3) Maximum Influence Out-Arborescence:* For a graph G, an influence threshold $\theta$, the maximum influence out-arborescence of a node u $\in$ V, MIOA(G,u,$\theta$), is defined as

$$MIOA(\mathcal{G}, u, \theta) = \bigcup_{v \in V, p(MIP(\mathcal{G},u,v)) \geq \theta} MIP(\mathcal{G}, u, v).$$

Fig. 14. Equation for MIOA

By tuning the parameter $\theta$, influence regions of different sizes can be obtained. For a single node, its MIOA is clearly

a tree. For multiple seed nodes, we build upon the idea of local influence region and propose two algorithms

### A. Construction of DAG 1

We observe that any DAG has at least one topological ordering. Conversely, given a topological ordering, if only edges going from a node of low rank to one with high rank are allowed, the resulting graph is a DAG.

To obtain the topological ordering given seed set S, we first introduce a (virtual) super root node R that is connected to all seed nodes with edge probability. We follow the following steps:

1) Let $G_R$ = ($V_{GR}, E_{GR}$) where $V_{GR}$ = V ∪ R and $E_{GR}$ = E∪(R,$S_k$)|∀$S_k$ ∈ S.
2) We build MIOA($G_R$,R,$\theta$) by calculating a Dijkstra tree from R
3) After removing R and its edges from MIOA($G_R$,R,$\theta$), we obtain a singly connected DAG $D_1$ = ($V_{D1}$,$E_{D1}$) on which BP algorithms can be directly applied and used to estimate the influence spread from S
4) However, $D_1$() is very sparse (with n-k edges) since many edges are removed.
5) We then augment $D_1$() with additional edges. Note that MIOA($G_R$,R,$\theta$) provides a topology ordering
6) Rank grows as the node is further away from R. We include in $D_1$() all edges in G whose end points are in $D_1$() and go from a node with lower rank to one with higher rank. Clearly, the resulting graph is a DAG

**Algorithm 2:** Calculate $\mathcal{D}_1(S)$ from a seed set $S$

**input** : $\mathcal{G}, S, \theta$

Build $\mathcal{G}_R = (V_{\mathcal{G}_R}, E_{\mathcal{G}_R})$
$\mathcal{D}_1(S) = MIOA(\mathcal{G}_R, R, \theta) \backslash R$
Calculate $r(v), \forall v \in V_{\mathcal{D}_1}$ (Eq. (6))
**foreach** $(u, v) \in V_{\mathcal{G}_R}$ **do**
  **if** $r(u) < r(v)$ *and* $(u, v) \in E$ **then**
    $\mathcal{D}_1(S) = \mathcal{D}_1(S) \cup (u, v)$

**output**: $\mathcal{D}_1(S)$

## XIII. Conclusion

Thus, we have studied information propagation models and the influence maximization problem. We have also studied various methods implemented to resolve the influence maximization problem like the Greedy algorithm and DAGs.


## Acknowledgment

I would like to express my gratitude and appreciation to all those who gave me the opportunity to complete this report. A special thanks to our guide, **Prof.Reena Pagare**, whose help, stimulating suggestions and encouragement helped me to coordinate my seminar.


## References


[1] W. Chen, Y. Wang, and S. Yang, Efficient influence maximization in social networks, in Proc. KDD, New York, NY, USA, 2009, pp. 199208.

[2] D. Kempe, J. M. Kleinberg, and . Tardos, "Maximizing the spread of influence through a social network" In Proceedings of the 9th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, pages 137146, 2003.

[3] Stanley Wasserman, Katherine Faust, Social Network Analysis Methods and Applications Part of Structural Analysis in the Social Sciences , University of Illinois, Urbana-Champaign, University of South Carolina, November 1994, pp. 1-116

[4] Miles Walker, August 2011, The History of Social Networking,[Online] Available: http://www.webmasterview.com/2011/08/social-networking-history/

[5] B. Liu, G. Cong, D. Xu, and Y. Zeng, Time constrained influence maximization in social networks, in Proc. ICDM, Washington, DC, USA , 2012, pp. 439448.

[6] J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. VanBriesen, and N. S. Glance. Cost-effective outbreak detection in networks. In Proceedings of the 13th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, pages 420429, 2007.

[7] E. Cohen. Size-estimation framework with applications to transitive closure and reachability. J. Comput. Syst. Sci., 55(3):441453, 1997.

[8] D. Coppersmith and S. Winograd. Matrix multiplication via arithmetic progressions. J. Symb. Comput., 9(3):251280, 1990.

[9] Chen, W., Wang, C., Wang, Y.: Scalable influence maximization for prevalent viral marketing in large-scale social networks. In: Proceedings of KDD 10, New York, NY, USA, ACM (2010) 10291038

[10] Yedidia, J.S., Freeman, W.T., Weiss, Y. In: Understanding belief propagation and its generalizations, San Francisco, CA, USA (2003) 239269