2012 IEEE 12th International Conference on Data Mining Workshops

Maximizing Information Spread Through Influence Structures in Social Networks

Saurav Pandit* †, Yang Yang*, Nitesh V. Chawla*
*Dept. Computer Sc. & Engg.
University of Notre Dame
Notre Dame, IN 46556, USA
[spandit, yyang1, nchawla]@nd.edu
†Intent Media
180 Varick St. Ste 936
New York, NY 10014, USA
saurav.pandit@intentmedia.com

***Abstract*—Finding the most influential nodes in a network is a much discussed research topic of recent time in the area of network science, especially in social network analysis. The topic of this paper is a related, but harder problem. Given a social network where neighbors can influence each other, the problem is to identify k nodes such that if a piece of information is placed on each of those k nodes, the overall spread of that information (via word-of-mouth or other methods of influence flow) is maximized. The amount of information spread can be measured using existing information propagation models. Recent studies, which focus on how quickly k high-influential nodes can be found, tend to ignore the overall effect of the information spread. On the other hand some legacy methods, which look at all possible propagation paths to compute a globally optimal target set, present severe scalability challenges in large-scale networks. We present a simple, yet scalable (polynomial time) algorithm that outperforms the existing state-of-the-art, and its success does not depend significantly on any kind of tuning parameter. To be more precise, when compared to the existing algorithms, the output set of k nodes produced by our algorithm facilitates higher information spread – in almost all the instances, consistently across the commonly used information propagation models. The original algorithm in this paper, although scalable, can have higher running time than some standard approaches, e.g. simply picking the top k nodes with highest degree or highest PageRank value. To that end, we provide an optional speedup mechanism that considerably reduces the time complexity while not significantly affecting the quality of results vis-a-vis the full version of our algorithm.**

***Keywords*-influence propagation, social networks, viral marketing**

I. Introduction

Locating the influence hubs within a population is an interesting and relevant problem – whether in the context of designing effective political or marketing campaigns, understanding what triggers *viral* spread of media, trends or opinions, or in the domain of disease control and epidemiology. While various versions of this problem have been studied for almost a decade now, the advent of very large scale social networks, growing influence of social media, and overall, the increasing volatility, plurality and dynamism of the process of influence formation have introduced unprecedented complexities to this problem. The sheer increase in size of the instances of these problems renders many previously accepted optimal or near-optimal solutions to be almost useless (on the account of scalability) and motivates us to seek improved, scalable solutions.

Besides the added complexity or scale of the problem, what makes it further investigation-worthy is the exploding reliance on viral marketing as a promotional tool, which is now-a-days adopted by businesses, political groups and humanitarian organizations alike. Research [1] confirms that people trust the words of those in their close circles (family, friends etc.) more than others. Hence, it is no surprise that many now firmly believe that viral marketing strategy is far more effective than advertising on any other media outlet [1], [2]. However, contrary to previous research and general intuition, a recent study [3] involving almost half a billion Facebook users has shown significant effect of weak ties in information propagation. We will discuss more about that later. But this phenomenon motivates that, if possible, looking at propagation via *all* edges should be preferred over working with small amount of local information.

Now let us consider a real-life scenario where a small company develops a product (e.g., a smartphone application or an "app"). Because of limited resources, they decide to give this product for free to a small number of *chosen* people and rely on the word-of-mouth to spread. How should these (say, k number of) people be chosen? This was the basis of the *influence maximization* problem as it was originally proposed by Kempe, Kleinberg and Tardos [4]. However, the problem we consider is not simply picking the k most influential people, but a target set of k people such that *overall influence spread is maximized*. We elaborate more on this important distinction throughout the rest of this paper.

We represent social networks as graphs, where nodes are people/entities and each edge represents some form of relationship/interaction between two people/entities. This means that our algorithm simply requires a static graph representation of the social network instead of any list of temporal events or data streams. There can be multiple edges

978-0-7695-4925-5/12 $26.00 © 2012 IEEE
DOI 10.1109/ICDMW.2012.140

between two nodes, representing different types of relationships. Note that not all graph representations of social networks are indicative of influence structures. For example, in a citation network, an article A can be considered to be influenced by article B, if B is cited by A. Similarly, in a Twitter network, a "retweet" can be considered as a measurable unit of influence exerted by the author of the original tweet. However, edges in a simple friendship or contact network may not necessarily imply influence. In this paper, when we say "social network", we often allude to an influence structure embedded on the social network, i.e. we work with graphs where edges represent dyadic relationships that are, in some way, indicative of influence.

A. Our Contribution

We present a simple and scalable algorithm that does not depend on any "tuning" parameter. It takes as input a graph $G = (V, E)$, V and E being the set of nodes and edges respectively. It also takes as input an integer k and finally outputs a set $T \subset V$, known as the *target set*, where $|T| = k$. For the remainder of the paper, we denote $|V| = n$ and $|E| = m$. We show that under the information cascade models, this target set consistently results in influence propagation as compared to the previously known scalable methods. We define $V' \subseteq V$ to be the *seed set*, i.e. nodes in V' are the only nodes that can be selected into the target set T. In the complete version of our algorithm $V' = V$. We also offer faster versions of our original algorithm using a speed-up parameter c, where the V' is shrunk or pruned from V prior to running our algorithm. We also show that it is possible to decrease the running time of our algorithm using the speedup mechanism without significant loss in quality of the output. We will further draw specific contrasts to the existing algorithms as we discuss them in Section I-B.

We take a combinatorial optimization approach to this problem. We model the social network as a graph and then reduce the problem to efficiently finding a near-optimal solution to a weakly NP-complete problem. Our goal is two-fold: provide an elegant algorithm that improves on quality of results while effectively scaling to large networks. The application of our work is not for real-time or near-instantaneous prediction (akin to PageRank) but rather a batch process – a more appropriate scenario for marketing, viral spread, discovery of influential nodes in a social graph model. Our work fills an important and critical gap of high quality results and improved scalability (thus running time) over methods that provide quality results of similar calibre. Note the result quality is defined in terms of the total spread of information originating from the selected set of k influential nodes. A number of algorithms might produce results of high quality but become quickly intractable with modern networks scaling to millions and millions of nodes.

Further, a recent large-scale field experiment [3] by Bakshy et al. shows even though stronger ties are individually significant in influence propagation, the *aggregated* effect of abundant weak ties are responsible for propagation of higher majority of new information. Hence, we believe that it is imperative for any effective influence tracking algorithm to consider weak ties and long-range paths alongside the strong ties, which brings the issues of computational complexity and quality of results to the fore.

Finally, one of the most important distinction of our algorithm with the existing ones is that we look for overall maximization of influence spread, instead of just picking the top k influential nodes (which may have overlapping influence bases). This is confirmed in some of our experimental results. For example, greedily picking k nodes with the highest PageRank value performs poorly in `condmat` dataset for this very reason.

B. Related Work

The idea of assigning each customer (of a business) a value based on their influence on other customers was first introduced by Domingos and Richardson [5], [6] in 2001-02. They called it the customer's *network value*. Their work established the importance of viral marketing and how its success depends on identifying influential users. This motivated Kempe et al. [4] to formulate the influence maximization problem as a discrete optimization problem in 2003. Although their paper is considered a seminal work in this area, their proposed greedy algorithm lacks scalability. In recent years, there have been several attempts to address this issue.

Equating *centrality* of a node with its "importance", one can simply choose the the nodes with highest centrality values (e.g. betweenness centrality) or even with highest degrees. This process, while simple, usually fails to produce very high quality target sets. We think that this can be partly attributed to the fact that due to high assortative mixing in social networks, many high degree nodes are adjacent to other high degree nodes. This phenomenon extends to the nodes with the highest centrality values as well. Computing PageRank [7] is fast, but if we simply pick the nodes with highest PageRank value, it also suffers the same way as there are no guarantees that they are well dispersed across the network. This points to the fundamental problem with the greedy approach in this scenario and why simply trying to refine the greedy approach in [4] may fall short - as observed in [8].

Recently several other approaches have been proposed. One of the notable ones is a lazy-forward optimization technique [9] that performs well but not tractable for large networks. The known heuristic approaches that run fast include [10] and [11]. The algorithm in [10] looks at shortest path based influence cascading. However, as pointed out in [11], shortest paths are not necessarily related to propagation probabilities. Experimentally [11] is shown to be best known result so far, which uses *maximum influence paths*

instead of shortest paths. However, it seems the success of their algorithm is dependent, at least partially, on choosing a proper value for a tunable parameter called *influence threshold*. Additionally our experiments show some lack of consistency in its performance, especially if we assume the Linear Threshold (LT) model of information propagation (described in Section II).

II. Validation

Before describing and analyzing out algorithm in details, it will be helpful to describe our validation approach. After k people are chosen from a social network (by our or any algorithm), how can we actually evaluate their overall influence? Or how do we quantify diffusion of influence? We use existing models of information propagation to evaluate how effective the chosen set of k people is. These can be thought of as models of behavior of the spread of word-of-mouth in social networks. In Section II-A we give brief overviews of all of these propagation models.

A. Information Propagation Models

The *influence maximization* problem was originally proposed in 2003 by Kempe, Kleinberg and Tardos [4] as finding the k most influential nodes in a social network under some stochastic cascade models, such as the *independent cascade* (IC) model or the *linear threshold* (LT) model. They further generalized these models subsequently [12]. Several similar generalizations were investigated by others but at the core of which were the IC or LT model. Along with these two core models, we also validate our algorithms in two of the more elaborate and widely accepted models – *weighted cascade* (WC) model and the *trivalency* (TRI) model. We compare the target set generated by our algorithm to the target sets generated by the existing algorithms in these four commonly used models — IC, LT, WC and TRI — of influence propagation. Under these models, the norm is to call a node *active* once it is affected (adopts the product or acquires the information) as a result of another node's influence. A node that becomes active also becomes "contagious", i.e. it can now influence (and possibly activate) other nodes and thus information is diffused/cascaded throughout the network. We start by activating the k nodes chosen by any algorithm and then simulate subsequent node activations based on the propagation models to compare the spread.

In the following, we use the notation $\Gamma(v)$ to denote the neighborhood of a node v, $d(v)$ denotes the *weighted* in-degree of node v and $wt(u, v)$ denotes the weight of the (directed) edge (u, v). Note that $wt(u, v)$ is simply seen as the number of parallel edges from u to v. Also, all the parallel edges are counted individually in $d(v)$, i.e.

$$d(v) = \sum_{u \in \Gamma(v)} wt(u, v).$$

Threshold Models: In the these models a threshold value is assigned to (or picked by) each node which drives their decision making [13], [14]. We will describe the LT model, which is the most frequently used information propagation model. In this model a node v is influenced by each neighbor u with a probability factor $p_{uv} = \frac{wt(u,v)}{d(v)}$. Each node v chooses a threshold θ_v uniformly at random from the interval $[0, 1]$. The node v becomes active as soon as the following condition is satisfied:

$$\sum_{u \in \Gamma(v), u \in active} p_{uv} > \theta_v.$$

It is easy to see that if all the neighbors of v becomes active the LHS of the above expression equals 1. Hence, in the very small possibility that $\theta_v = 1$, v can be thought of as "immune", i.e. v will never get activated.

Cascade Models: In cascade models [15], [16], unlike in threshold models, whether u can influence v depends *directly* on a probability factor p_{uv}, i.e. once u is active, it activates $v \in \Gamma(u)$ with probability p_{uv}. Also unlike in threshold models, here each active node gets only one chance (per edge) to activate a neighbor. In the event that none of u's neighbors are activated, that branch of the cascade stops. The probability p_{uv} is defined differently in different cascade models. The simplest of such models is the IC model, where a uniform activation probability p is assigned to each edge across the network. In different trials, we set this p value to 1% or 10%. It is to be noted that u can activate v with probability p via each parallel edge between u and v. Hence, in the IC model,

$$p_{uv} = 1 - (1 - p)^{wt(u,v)}.$$

In the WC model we set p for each node v to be $1/d(v)$. Hence,

$$p_{uv} = 1 - (1 - \frac{1}{d(v)})^{wt(u,v)}.$$

In the TRI model, p is selected at uniformly at random from the set $\{0.1, 0.01, 0.001\}$ which corresponds to high, medium and low influence levels.

III. Our Algorithm

Now that we have described how are results were validated, and established a standard benchmark framework as used by other published methods, let us bring the focus back on our proposed algorithm. We are given $G = (V, E)$, a graph representing the network. As usual, V and E represents the set of nodes and edges with $|V| = n$ and $|E| = m$. Our goal is to find the k most influential nodes in this network. We use an optional speed-up parameter c, with $c \in [1, n/k]$. Lower c is, faster the algorithm runs and $c = n/k$ represents the full version of the algorithm (i.e. without speedup).

The algorithm is divided into two phases: `Learning` and `Search`. We will describe each phase separately. Let,

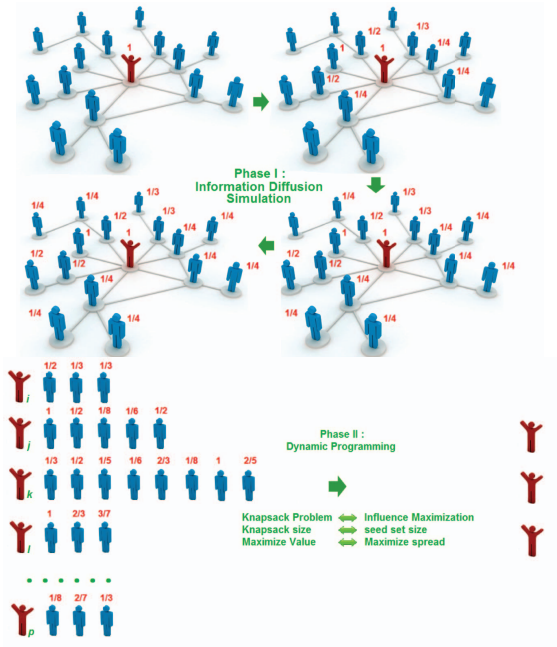

Figure 1. Algorithm overview. **(a)** Phase 1 of Diffusion$_{Complete}$. In the figure, $\alpha = 1$. **(b)** Phase 2. An $n \times n$ influence matrix is generated from the diffusion process in Phase 1. The matrix is run through a knapsack-type process for choosing k rows using dynamic programming.

$V' \subseteq V$ be the base set within which we search for the k nodes (target set) that exert most influence over V. Let V' be called the seed set. As mentioned before, in the full version of our algorithm (Diffusion$_{Complete}$), $V' = V$. In a speed-up version with speed-up parameter c, $|V'| = c \cdot k$. A speedup version of the algorithm with speed-up parameter c is denoted as Diffusion$_c$.

In the `Learning` phase, unit pieces of distinct information is placed on each node and allowed to diffuse under rules described in Algorithm 1. Some of the philosophies of this diffusion process are same as those used in PageRank [7] algorithm. For example, the likelihood of propagation of information depends on how many other people the target person is listening to, i.e. the in-degree of a node. We call this the "plurality" effect. The concept of information "fading" is also used by PageRank and well accepted by others. Throughout this diffusion process, we record the propagation of each piece of information and output them in the form of an *influence matrix*: $\mathbf{M}_{ck \times n}$. This matrix is then passed on to the next phase (`Search`).

In the matrix $\mathbf{M}$, each row i states the influence of node i on other nodes in the network. We can also see the i^{th} row of the matrix $\mathbf{M}$ as an *influence vector* of node i, denoted as $\vec{v}_i$. Let us define the combined effect of two such vectors, $\vec{v}_i$ and $\vec{v}_{i'}$, as

$$\vec{v}_i \bigoplus \vec{v}_{i'} = (max(\vec{v}_i[1], \vec{v}_{i'}[1]), \ldots, max(\vec{v}_i[n], \vec{v}_{i'}[n])).$$

Let us call this the *cumulative influence* of nodes i and i'.

Algorithm 1 Diffusion$_c$: Phase 1 (`Learning`)

Input: $G \leftarrow$ input graph. $G = (V, E)$.
Input: $k \leftarrow$ size of the target set.
Param: $c \leftarrow$ speedup factor. Note that, in case of Diffusion$_{Complete}$, $c = n/k$.
Initialize: $\alpha \leftarrow$ decay factor, traditionally set to 0.85.
Initialize: $\sigma_{ij} \leftarrow 0, \forall i, j \in V$.
Initialize: $V' \leftarrow$ the top $c \cdot k$ nodes with the highest PageRank or Degree values, picked greedily.
for $\forall u \in V'$ **do**
  $BFS(u) \leftarrow$ BFS tree from i with levels P_0 to P_h. Note, $P_0 = \{u\}$.
  If $v \in \Gamma(u)$, set $\sigma_{uv} \leftarrow \frac{\alpha}{d(v)}$.
  $\forall w \in \Gamma(v)$ and $\forall l$, $l = 0, 1, \ldots, h$ such that $\forall v \in P_l$ and $w \in P_{l+1}$, set $\sigma_{uw} \leftarrow \max_v [\frac{\alpha}{d(w)} \cdot wt(v, w) \cdot \sigma_{uv}]$.
end for
Output: $\mathbf{M} \leftarrow [\sigma_{ij}]$

Let us also define $value(\vec{v}_i) = \sum_{j=1}^{n} \vec{v}_i[j]$, which is simply the sum of all the vector elements, i.e. the total influence of i on all nodes. In Algorithm 2 we have $c.k$ *candidate nodes* (we select most influential nodes from these candidate nodes) and their corresponding influence vectors, which are represented by matrix $\mathbf{M}$. The problem is to select a set S of k nodes from the candidate nodes, such that $value(\bigoplus_{i \in S} \vec{v}_i)$ is maximized. Let us denote this problem to be $\mathcal{P}$.

Note that, by now we have reduced the influence maximization problem to this problem $\mathcal{P}$ on the matrix $\mathbf{M}$ generated in Phase 1. $\mathcal{P}$ is weakly NP-complete (akin to the Knapsack problem). In Phase 2, we seek an approximate solution of this problem using dynamic programming technique. The $ck \times n$ influence matrix M generated in Phase 1 is employed as Phase 2's input, where $M[i][j]$ represents the influence of node i on node j. The output is the set S containing the indices of k rows of $\mathbf{M}$, which corresponds to the k nodes in V' that maximizes the information spread. The implementation of the dynamic programing can be seen in the pseudocode (Algorithm 2).

In Algorithm 2, similar to standard dynamic programming setting, we use two bookmarking matrices: **influences** is an $(l + 1) \times (k + 1)$ matrix where each element is an influence vector; and **traces** is an $l \times k$ matrix where each element is a boolean value. The element $influences[x][y]$ is the cumulative influence vector of the optimal set of size y selected from the first x nodes. We first initialize all elements of first row and first column to be empty vectors in matrix $influences$; while for matrix $traces$ all elements are initialized to be $null$.

It is to be noted that the problem $\mathcal{P}$ is actually more complex than the standard Knapsack problem. For example, consider we have an optimal set S' selected from first $x - 1$

nodes, whose cumulative influence vector have high probability to activate a node x; in such case including node x into the optimal set S will be redundant. We use an *activation threshold*, denoted θ_c, to address this issue and control the redundancy. In Algorithm 2 line 11, if node x already has high probability to be activated by previously selected nodes (their cumulative influence vector is denoted $\vec{v}_{x-1,y}$ for convenience, which is equal to $influences[x-1][y]$), we do not consider it for the optimal set. Otherwise we decide whether to include it in line 12 to 18.

The first step of Algorithm 2 (lines 6 to 24) is to find out the set of k vectors which have maximum $value(\bigoplus_{i \in S} \vec{v}_i)$ as defined above. The second step (line 25 to 34) is a standard backtracking procedure to identify these k node ids.

Algorithm 2 Diffusion$_c$: Phase 2 (`Search`)

1: **Input:** $\mathbf{M} \leftarrow$ an $l \times n$ influence matrix produced by Algorithm 1, where $l = c.k$
2: **Input:** $k \leftarrow$ size of the target set
3: **Param:** $\theta_c \leftarrow$ activation threshold, set to 0.9
4: **Initialize:** $influences \leftarrow$ an $(l+1) \times (k+1)$ matrix
5: **Initialize:** $traces \leftarrow$ an $l \times k$ boolean matrix
6: **for** $x := 1$ to l **do**
7: **for** $y := 1$ to k **do**
8: $\vec{v}_{x-1,y}$:= $influences[x-1][y]$
9: $\vec{v}_{x-1,y-1}$:= $influences[x-1][y-1]$
10: $\vec{temp} := \vec{v}_x \bigoplus \vec{v}_{x-1,y-1}$ // Note, $\vec{v}_x = M[x]$
11: **if** $\vec{v}_{x-1,y}[x] < \theta_c$ **then**
12: **if** $value(\vec{v}_{x-1,y}) < value(\vec{temp})$ **then**
13: $influences[x][y]$:= $\vec{temp}$
14: $traces[x][y]$:= $true$
15: **else**
16: $influences[x][y]$:= $influences[x-1][y]$
17: $traces[x][y]$:= $false$
18: **end if**
19: **else**
20: $influences[x][y]$:= $influences[x-1][y]$
21: $traces[x][y]$:= $false$
22: **end if**
23: **end for**
24: **end for**
25: // Backtracking to collect optimized set
26: $number := k$
27: $S := null$
28: **for** $x := l$ to 1 **do**
29: **if** $traces[x][number]$ **then**
30: $S \leftarrow S \cup \{x\}$
31: $number := number - 1$
32: **end if**
33: **end for**
34: **Output:** S

A. Time Complexity

Lemma 1: The complexity of the `Learning` phase for the Diffusion$_{Complete}$ algorithm is $O(n(n+m))$.

Proof: To analyze the running time of Phase 1, let us consider Algorithm 1. For Diffusion$_{Complete}$, $c = n/k$. Hence $|V'| = |V| = n$. Note that in such a case, Line 6 need not be executed. The for-loop starting in Line 7, is executed n times. And the pseudocode inside the for-loop (Lines 8-10) constructs a breadth-first search (BFS) tree. The details of the BFS procedure is omitted, but a standard implementation takes $O(n+m)$ rounds. ■

Lemma 2: The complexity of the `Search` phase for the Diffusion$_{Complete}$ algorithm is $O(k^2 n)$.

Proof: To analyze the complexity of Phase 2, let us refer to the pseudocode in Algorithm 2. For Phase 2, If we assume the pseudocode from line 8 to line 19 as a unit operation, then it is executed $k \cdot ck$ times. Within that part of the code, we are comparing two vectors whose maximum size is $|V| = n$, which can be trivially done in $O(n)$. Additionally from line 28 to line 33 the output set is constructed by a for-loop, which executes ck times. Hence, overall running time for Phase 2 is $O(k \cdot ck \cdot n + ck)$. Diffusion$_{Complete}$, $ck = n$. Hence, Phase 2 for $Diffusion_{Complete}$ is $O(k^2 n)$. ■

Note that even though the worst case complexity depends on the number of columns in the influence matrix, i.e. n, in practice it turns out to be much smaller by a enforcing a simple, practical screen. In our experiments, we ignore any σ_{ij} value less than 10^{-4}. This does not affect the quality of the algorithm at all, but significantly reduces runtime.

Theorem 3: The Diffusion$_{Complete}$ algorithm runs in polynomial time.

Proof: Follows directly from Lemmas 1 and 2. ■

B. Speed-up

The following Corollary presents a more general version of Lemmas 1 and 2 and formally describes the effects of the speed-up procedure. As seen in the experimental results (See Figure 2, other results not shown here for space constraints), the speedup versions perform quite well. There is clearly a trade-off between speed and quality. However, the key point is that running time can be decreased considerably without significantly comprising the quality. Choosing $c = n/k$ represents our original algorithm. Choosing $c = 1$ represents simply picking the k top nodes with highest PageRank or degree (Line 6 of Algorithm 1).

Corollary 4: The speed-up mechanism reduces the complexity of Phase 1 to $O(k(m+n))$ and Phase 2 to $O(k^2 n)$, where the speed-up factor $c \in [1, n/k)$ is a constant within the Big-O notation.

Note that the speed-up operation is optional. As seen in Figure 3, the full version of the algorithm (Diffusion$_{Complete}$) has a higher running time as compared to some of the other algorithms. But it scales to large networks. *And most importantly, as we will see in the next*

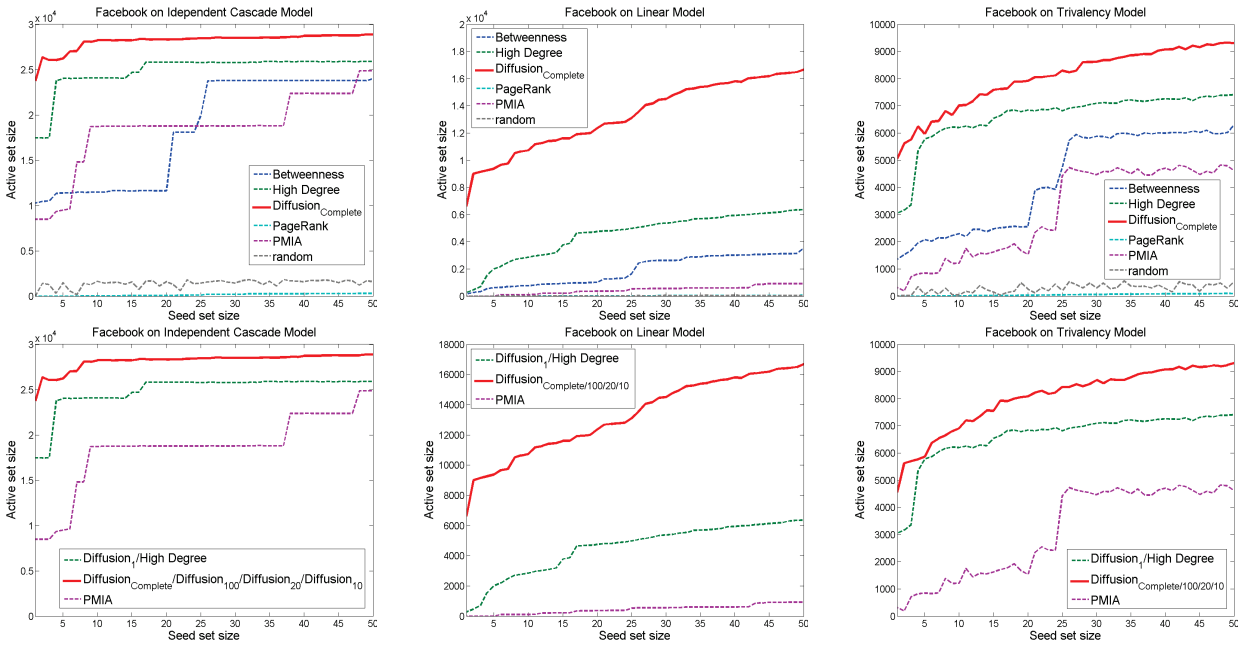

Figure 2. Results in the Facebook New Orleans (`facebook`) dataset. Size of the target set (1 to 50) on the X-axis. Number of activated nodes on the Y-axis. **Above:** The reachability of our algorithm as compared to other existing algorithms over seed sets of various size. The plots are respectively in IC, LT and TRI models. **Below:** Results in the same respective models comparing our original result (Red) with the speed-up versions and PMIA (Purple). Note that the Cyan line represents speed-up factor 1, which is equivalent to the greedily picking the highest degree nodes. (Color online)

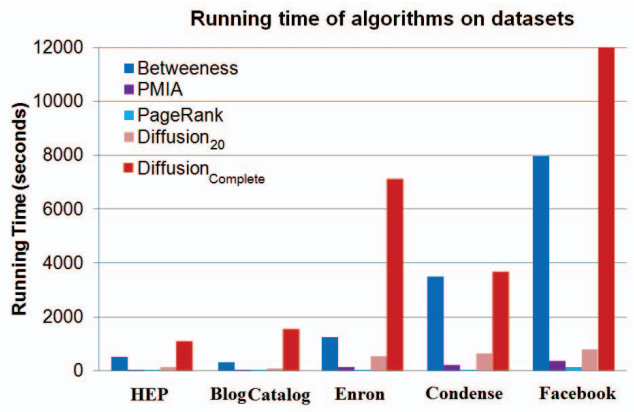

Figure 3. Running time comparison, from the smallest to the largest networks (Color online)

section, it consistently and significantly outperforms the existing algorithms in terms of information spread. As an example, let us consider the results in the `facebook` dataset ($m = 817,090$ and $n = 63,731$), where the algorithm takes little more then 3 hours, which we consider reasonable time given the superior quality of the output. However, note that the speedup version ($c = 20$) runs in comparable time as the faster algorithms and still produces better results. Again, in the `Facebook` dataset with $k = 50$, the spread of information in the LT model by Diffusion_{20} is about 3 times larger than picking the top 50 nodes with highest degree.

IV. Experimental Results

The experiments are conducted with two major objectives: 1) performance comparison of the complete version of our algorithm with existing algorithms; 2) establish the effectiveness of our speed-up mechanism. Although running-time is not the focus of our work, we keep track of it in order to show that our algorithm is scalable. The experiments were run on a machine with Intel(R) Core(TM) i5 CPU @ 3.0GHz and 8G memory. The source code and data are made available at: http://www.nd.edu/~dial/share/Diffusion.zip for reproducibility.

A. Benchmarks

We compare our $\text{Diffusion}_{Complete}$ algorithm against several existing algorithms. The simplest strategy for choosing k most influential nodes is to pick the k nodes with highest out-degree (`High Degree`). Similar, but slightly more involved methods include picking the top k nodes with highest `Betweenness` values or `PageRank` values. We also compare it against the best known result so far, which is the `PMIA` algorithm [11]. We received the PMIA executable code from the authors (due to the proprietary nature of it, they could not provide us with the source code). For benchmarking purposes, we also present the results if we pick k nodes randomly (`random`).

All the benchmarks used the validation models described in section II.

The speed-up versions (Diffusion$_c$), for c =10, 20 and 100, are also compared with Diffusion$_{Complete}$, Diffusion$_1$ and PMIA. Note that Diffusion$_1$ is equivalent to `High Degree` (or `PageRank`).

B. Datasets

Following are the six datasets we use for experimentation. Each dataset can be represented as a network.

Co-authorship Networks: We have used three co-authorship networks in the fields of Network Science (`netsci`), High Energy Physics (`hept`) and Condensed Matter (`condmat`). Co-authoring papers can undoubtedly be considered as events where co-authors, in some way, influence each other. The `netsci` dataset [17] was compiled by M. Newman in 2006. It contains 1,589 nodes and 5,484 edges. He also complied the `condmat` dataset [18] that we use. It is a much larger network than `netsci` with 16,726 nodes and 95,188 edges. The third collaboration network we use is relatively small, with 9,875 nodes and 51,971 edges. The `hept` dataset [19] describes the collaboration network based on arXiv's High Energy Physics Theory category.

Communication Networks: The Enron email communication network (`enron`) covers all the email communication within a dataset of around half million emails. This data was originally made public, and posted to the web, by the Federal Energy Regulatory Commission during its investigation. It contains 22,477 nodes and 164,081 edges.

Online Social Networks: We use two online networks representing friendship among people. BlogCatalog (`blog`) [20] is the social blog directory which manages the bloggers and their blogs. The edges in this network represent the friendship among the bloggers. It contains 10,312 nodes and 333,983 edges. The Facebook dataset (`facebook`) [21] contains user-to-user links from the Facebook New Orleans networks, containing 63,731 nodes and 817,090 edges.

C. Observations

We describe the key observations by referring to the results in the `facebook` dataset (Figure 2). Note that this is only a representative result. Our results are consistent across all the six networks we experimented on. Please contact the authors for the full set of results.

As can be seen Figure 2, our algorithm outperforms the existing algorithms across all Information Cascade Models and for all target set sizes (1 to 50). We also see that the speed-up versions outperform the existing algorithms for even small c (10, 20 etc.). In fact, as seen in the IC and LT model, the difference between Diffusion$_{Complete}$ and Diffusion$_{10}$ or Diffusion$_{20}$ is so small that the plots basically overlap each other. Another interesting thing to note is the large improvements by increasing c from 1 to 10 or 20. Note that $c = 1$ means we simply pick the top k node with highest degree. Instead, if we pick a seed set of size $10k$ and run them through our algorithm, we get far better results.

The most remarkable observation is the consistency in the performance of our algorithm. As can be seen in Figure 2 and all the other results (removed from this version due to space constraints), if we assume the LT model of information propagation, the performance of PMIA seems to suffer significantly as compared to in other models. Picking the highest PageRank value nodes gives decent target sets in many datasets, but seems to collapse in the online social networks (`facebook` and `blogs`). Diffusion$_{Complete}$ shows a superior performance for all the six networks, in all four propagation models and for almost all k values.

V. Conclusion

In summary, we present an novel and efficient way to take an exhaustive look at the possible information propagation paths in a social network, which allows us to determine the most influential nodes better than the existing algorithms. The algorithm we present is scalable and its success does not depend significantly on any kind of tuning parameter. We also provide an optional speed-up mechanism (which uses a speed-up parameter).

Note that we only look at a static snapshot of the network. We believe the results can be improved if further attention is paid on the different types of edges (relationships or interactions) that occur in networks. For example, in `facebook`, we treat all the edges the same. Whereas in reality, friendship, profile visit, messages etc. can imply different degrees of influence. This also highlights the necessity of developing comprehensive influence metrics for such online social networks. The issue here is that such metrics will be characteristic of the social network itself, and will have to be developed on a case by case basis. However, if such metrics are present, influence maximization algorithms will not have to rely on information propagation models for testing.

Acknowledgement

This research was sponsored in part by the Army Research Laboratory and was accomplished under Cooperative Agreement Number W911NF-09-2-0053 and in part by the National Science Foundation (NSF) Grant BCS-0826958.. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

REFERENCES

[1] J. Nail, *The Consumer Advertising Backlash*. Forrester Research and Intelliseek Market Research Report, 2004.

[2] I. R. Misner and V. Devine, *The world's best-known marketing secret: building your business with word-of-mouth marketing*. Bard Press, 1999.

[3] E. Bakshy, I. Rosenn, C. Marlow, and L. Adamic, "The role of social networks in information diffusion," Tech. Rep. arXiv:1201.4145, Jan 2012.

[4] D. Kempe, J. Kleinberg, and E. Tardos, "Maximizing the spread of influence through a social network," in *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, ser. KDD '03, 2003, pp. 137–146.

[5] P. Domingos and M. Richardson, "Mining the network value of customers," in *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, ser. KDD '01, 2001, pp. 57–66.

[6] M. Richardson and P. Domingos, "Mining knowledge-sharing sites for viral marketing," in *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, ser. KDD '02, 2002, pp. 61–70.

[7] S. Brin and L. Page, "The anatomy of a large-scale hypertextual web search engine," *Computer Networks*, vol. 30, pp. 107–117, April 1998.

[8] W. Chen, Y. Wang, and S. Yang, "Efficient influence maximization in social networks," in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, ser. KDD '09, 2009, pp. 199–208.

[9] J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. VanBriesen, and N. Glance, "Cost-effective outbreak detection in networks," in *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, ser. KDD '07, 2007, pp. 420–429.

[10] M. Kimura and K. Saito, "Tractable models for information diffusion in social networks," in *Proceedings of the 10th European Conference on Principles and Practice of Knowledge Dicovery in Databases*, ser. PKDD '06, 2006, pp. 259–271.

[11] W. Chen, C. Wang, and Y. Wang, "Scalable influence maximization for prevalent viral marketing in large-scale social networks," in *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, ser. KDD '10, 2010, pp. 1029–1038.

[12] D. Kempe, J. Kleinberg, and Éva Tardos, "Influential nodes in a diffusion model for social networks," in *32nd International Colloquium on Automata, Languages and Programming*, ser. ICALP '05, 2005, pp. 1127–1138.

[13] M. Granovetter, "Threshold Models of Collective Behavior," *American Journal of Sociology*, vol. 83, no. 6, pp. 1420–1443, 1978.

[14] T. Schelling, *Micromotives and macrobehavior*, ser. Fels lectures on public policy analysis. Norton, 1978.

[15] J. Goldenberg, B. Libai, and Muller, "Using complex systems analysis to advance marketing theory development," *Academy of Marketing Science Review*, 2001.

[16] J. Goldenberg, B. Libai, and E. Muller, "Talk of the Network: A Complex Systems Look at the Underlying Process of Word-of-Mouth," *Marketing Letters*, pp. 211–223, Aug. 2001.

[17] M. E. J. Newman, "Coauthorships in network science: coauthorship network of scientists working on network theory and experiment," *Phys. Rev. E*, vol. 74, May 2006.

[18] ——, "The structure of scientific collaboration networks," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 98, no. 2, pp. 404–409, January 2001.

[19] J. Leskovec, J. Kleinberg, and C. Faloutsos, "Graph evolution: Densification and shrinking diameters," *ACM Trans. Knowl. Discov. Data*, vol. 1, no. 1, p. 2, 2007.

[20] R. Zafarani and H. Liu, "Social computing data repository at ASU," 2009. [Online]. Available: http://socialcomputing.asu.edu

[21] B. Viswanath, A. Mislove, M. Cha, and K. P. Gummadi, "On the evolution of user interaction in facebook," in *Proceedings of the 2nd ACM SIGCOMM Workshop on Social Networks (WOSN'09)*, 2009.