Efficient Influence Maximization in Social Networks

Wei Chen
Microsoft Research Asia
Beijing, China
weic@microsoft.com

Yajun Wang
Microsoft Research Asia
Beijing, China
yajunw@microsoft.com

Siyu Yang
Dept. of Computer Science
Tsinghua University
Beijing, China
siyu.yang@gmail.com

ABSTRACT

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. In this paper, we study the efficient influence maximization from two complementary directions. One is to improve the original greedy algorithm of [5] and its improvement [7] to further reduce its running time, and the second is to propose new degree discount heuristics that improves influence spread. We evaluate our algorithms by experiments on two large academic collaboration graphs obtained from the online archival database arXiv.org. Our experimental results show that (a) our improved greedy algorithm achieves better running time comparing with the improvement of [7] with matching influence spread, (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, and more importantly (c) the degree discount heuristics run only in milliseconds while even the improved greedy algorithms run in hours in our experiment graphs with a few tens of thousands of nodes.

Based on our results, we believe that fine-tuned heuristics may provide truly scalable solutions to the influence maximization problem with satisfying influence spread and blazingly fast running time. Therefore, contrary to what implied by the conclusion of [5] that traditional heuristics are outperformed by the greedy approximation algorithm, our results shed new lights on the research of heuristic algorithms.

Categories and Subject Descriptors

F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems

General Terms

Algorithms, Experimentation, Performance

Keywords

social networks, influence maximization, heuristic algorithms

KDD'09, June 28–July 1, 2009, Paris, France.

1. INTRODUCTION

Recently many large-scale online social network sites, such as Facebook and Friendster, become successful because they are very effective tools in connecting people and bringing small and disconnected offline social networks together. Moreover, they are also becoming a huge dissemination and marketing platform, allowing information and ideas to influence a large population in a short period of time. However, to fully utilize these social networks as marketing and information dissemination platforms, many challenges have to be met. In this paper, we present our work towards addressing one of the challenges, namely finding influential individuals efficiently in a large-scale social network.

Consider the following hypothetical scenario as a motivating example. A small company develops a cool online application for an online social network and wants to market it through the same network. It has a limited budget such that it can only select a small number of initial users in the network to use it (by giving them gifts or payments). The company wishes that these initial users would love the application and start influencing their friends on the social network to use it, and their friends would influence their friends' friends and so on, and thus through the word-of-mouth effect a large population in the social network would adopt the application. The problem is whom to select as the initial users so that they eventually influence the largest number of people in the network, i.e., the problem of finding influential individuals in a social network.

This problem, referred to as *influence maximization*, would be of interest to many companies as well as individuals that want to promote their products, services, and innovative ideas through the powerful word-of-mouth effect (or called viral marketing). Online social networks provide good opportunities to address this problem, because they are connecting a huge number of people and they collect a huge amount of information about the social network structures and communication dynamics. However, they also present challenges to solve the problem. The social networks are large-scale, have complex connection structures, and are also very dynamic, which means that the solution to the problem needs to be very efficient and scalable.

Domingos and Richardson [3, 8] are the first to study influence maximization as an algorithmic problem. Their methods are probabilistic, however. Kempe, Kleinberg, and Tardos [5] are the first to formulate the problem as the following discrete optimization problem. A social network is modeled as a graph with vertices representing individuals and edges representing connections or relationship between two individuals. Influence are propagated in the network according to a stochastic cascade model. Three cascade models, namely the independent cascade model, the weight cascade model, and the linear threshold model, are considered in [5]. Given a social network graph, a specific influence cascade model,

and a small number k, the influence maximization problem is to find k vertices in the graph (refered to as *seeds*) such that under the influence cascade model, the expected number of vertices influenced by the k seeds (referred to as the *influence spread* in the paper) is the largest possible.

Kempe et al. prove that the optimization problem is NP-hard, and present a greedy approximation algorithm applicable to all three models, which guarantees that the influence spread is within $(1 - 1/e)$ of the optimal influence spread. They also show through experiments that their greedy algorithm significantly outperforms the classic degree and centrality-based heuristics in influence spread.

However, their algorithm has a serious drawback, which is its efficiency. A key element of their greedy algorithm is to compute the influence spread given a seed set, which turns out to be a difficult task. Instead of finding an exact algorithm, they run Monte-Carlo simulations of the influence cascade model for sufficiently many times to obtain an accurate estimate of the influence spread. As a result, even finding a small seed set in a moderately large network (e.g. 15000 vertices) could take days to complete on a modern server machine.

Several recent studies aimed at addressing this efficiency issue. In [6], Kimura and Saito propose shortest-path based influence cascade models and provide efficient algorithms of compute influence spread under these models. However, since the influence cascade models are different, they do not directly address the efficiency issue of the greedy algorithms for the cascade models studied in [5].

In [7], Leskovec et al. present an optimization in selecting new seeds, which is referred to as the "Cost-Effective Lazy Forward" (CELF) scheme. The CELF optimization uses the submodularity property of the influence maximization objective to greatly reduce the number of evaluations on the influence spread of vertices. Their experimental results demonstrate that CELF optimization could achieve as much as 700 times speedup in selecting seed vertices, which is a very impressive result. However, our experiments show that the improved algorithm still takes a few hours to complete in a graph with a few tens of thousands of vertices, so it is still not efficient for large-scale networks.

In this paper, we tackle the efficiency issue of influence maximization from two complementary directions. In one direction, we design new schemes to further improve the greedy algorithm, and combine our scheme together with the CELF optimization to obtain faster greedy algorithms. In the other direction, we propose new *degree discount* heuristics with influence spreads that are significantly better than the classic degree and centrality-based heuristics and are close to the influence spread of the greedy algorithm. The biggest advantage of our heuristics is their speed, as they are many orders of magnitude faster than all greedy algorithms.

Our new greedy algorithms and degree discount heuristics are derived from the independent cascade model and weighted cascade model. We conduct extensive experiments on two real-life collaboration networks to compare our algorithms with the CELF optimized algorithm as well as classic degree and centrality heuristics. The two metrics we compare are influence spread and running time. For our new greedy algorithms, their influence spread exactly match with the original greedy algorithm, whereas their running times are 15% to 34% shorter than the CELF optimization. For our degree discount heuristics, their influence spread are close to that of the greedy algorithm, and always outperforms the classic degree and centrality-based heuristics. One particular heuristic tuned for the independent cascade model with a small propagation probability almost matches the influence spread of the greedy algorithm (same as the greedy algorithm in one experimental graph and 3.4% lower in another graph) in the independent cascade model. Their biggest advantage is their blazingly fast speed — they complete the task in only a few milliseconds, which is less than one-millionth of time of the fastest greedy algorithm. Furthermore, we also run our greedy and heuristic algorithms on the linear threshold model. Our results demonstrate that our new algorithms still have good influence spread performance in the linear threshold model. Therefore, they are robust across these models.

These results provide us a new perspective in the study of the influence maximization problem. Instead of focusing our effort in further improving the running time of the greedy algorithm, it perhaps more promising to focus on improving heuristics that could be a million times faster and making their influence spread close to the greedy algorithm.

To summarize, our contributions are mainly twofold. First, we provide further improvement to the greedy algorithm that has guaranteed approximate influence spread. Second, and more importantly, we propose new heuristics that have influence spreads close to the greedy algorithm while running at more than six orders of magnitude faster than the greedy algorithm. Encouraged by these results, we suggest that the promising approach in solving the influence maximization problem for large-scale social networks is to invest in heuristics to improve their influence spread, rather than trying to improve the running time of the greedy algorithms, which is the approach taken by most previous studies.

The rest of the paper is organized as follows. Section 2 presents the greedy algorithm and our new improvements to the greedy algorithm in the independent cascade model and the weighted cascade model. Section 3 presents the degree discount heuristics. Section 4 show our experimental results. We conclude the paper in Section 5.

2. IMPROVING THE GREEDY ALGORITHM

In this section, we discuss improvement of the greedy algorithm proposed by Kempe, et al. [5] for the *independent cascade* model as well as the *weighted cascade* model.

2.1 Problem definition and the greedy algorithm

A social network is modeled as an undirected graph $G = (V, E)$, with vertices in V modeling the individuals in the network and edges in E modeling the relationship between individuals. For example, in our experiment section, we study coauthorship graphs where vertices are authors of academic papers and two vertices have an edge if the two corresponding authors have coauthored a paper.[1] We use n to denote the number of vertices and m to denote the number of edges throughout the paper. For convenience, Table 1 lists important variables used throughout the paper.

Let S be the subset of vertices selected to initiate the influence propagation, which we call the *seed set*. Let $RanCas(S)$ denote 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 paper 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.

[1] Our coauthorship graphs studied in the experiment section are actually multigraphs, with parallel edges between two vertices denoting the number of papers coauthored by the two authors, same as in [5]. For simplicity, however, in our explanation of the algorithms and heuristics, we treat the graph as a simple graph. The results can be generalized to multigraphs in a straightforward way.

Table 1: Important variables used in the paper.

Variables	Descriptions
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 [1] used in Algorithm 3.
p	propagation probability in the IC model
d_v	degree of vertex v in G
t_v	number of neighbors of vertex v already selected as seeds

Algorithm 1 GeneralGreedy(G, k)

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

Algorithm 1 describes the general greedy algorithm given a random process $RanCas()$. In each round i, the algorithm adds one vertex into the selected set 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 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. Time complexity of all algorithms in the paper are summarized in Table 2 for convenience of comparison.

In [7], Leskovec et al. presents 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 has the same influence spread as the original greedy algorithm but is much faster, in fact 700 times faster as reported in [7]. In this paper, we compare the running times of our improved greedy algorithms with the CELF-optimized greedy algorithm and show that we can further improve the greedy algorithm.

Table 2: Time complexity of algorithms in this paper.

Algorithms	Time complexity
Algorithm 1: GeneralGreedy	$O(knRm)$
Algorithm 2: NewGreedyIC	$O(kRm)$
Algorithm 3: NewGreedyWC	$O(kRTm)$
Algorithm 4: DegreeDiscountIC	$O(k \log n + m)$

The main difference between cascade models is the random cascade process $RanCas(S)$, which will be explained below.

2.2 Improvement for the independent cascade model

In the *independent cascade* (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 other words, if there are ℓ neighbors of v that are in A_i, $v \in A_{i+1}$ with probability $1 - (1-p)^\ell$. 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 $v \in V \setminus S$, 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 $v \in V \setminus S$ 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.[2]

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 (a) they do not affect the estimate of each individual vertex, (b) 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 (c) 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. Our experiment results show that using the same number R NewGreedyIC achieves the same influence spread as GeneralGreedy, so there is no need to increase R to compensate the correlation effect.

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. How-

[2]In practice, the running time of $RanCas(S)$ is dependent on the size of the influenced vertices, which is much smaller than n. Therefore, the actual improvement on the running time is not as much as a factor of n, but is still significant.

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
```

ever, 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.

2.3 Improvement for the weighted cascade 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 $1-(1-1/d_v)^\ell$.

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 $\vec{uv}$ and $\vec{vu}$. We still use d_v to denote the degree of v in the original graph.

Consider the following random process: For each edge $\vec{uv} \in \hat{E}$, we remove it from $\hat{G}$ with probability $1-1/d_v$. Let $RanWC(\hat{G})$ denote this random process and $G' = RanWC(\hat{G})$ denote the resulting directed graph. Therefore, the output of the random cascade process $RanCas(S)$ in the WC model is the same as $R_{G'}(S)$, the set of vertices reachable in G' from S.

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 $O(n^{2.38})$ [2], 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 [1] for estimating the number of all reachable vertices from every vertex.

We now briefly explain Cohen's randomized algorithm in our context of computing $|R_{G'}(S \cup \{v\})|$ for all vertices $v \in V \setminus S$. 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 $v \in V$, let v^* denote the corresponding (collapsed) vertex in V^*. Let $S^* = \{v^* \in V^* \mid v \in S\}$. 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 $S \subseteq V$, Let $w(S^*) = \sum_{v^* \in S^*} w(v^*)$. One important property of G'^* with weights $w()$ is that $w(R_{G'^*}(S^* \cup \{v^*\})) = |R_{G'}(S \cup \{v\})|$.

Cohen's algorithm estimates $|R_{G'}(S \cup \{v\})|$ in T iterations. In the i-th iteration, on every vertex $v^* \in V^*$ we take a random sample $X^i_{v^*}$ according to an exponential distribution with the probability density function $w(v^*)e^{-w(v^*)x}$, $x \geq 0$ (one way to obtain the sample is to sample z uniformly from $[0,1]$ and output $-(\ln z)/w(v^*)$). Then we compute

$$Y^i_{v^*} = \min_{u^* \in R_{G'^*}(S^* \cup \{v^*\})} X^i_{u^*}.$$

Notice that each iteration only needs a constant number of traversals of graph G'^* and thus can be completed in $O(m)$ time. After the T iterations are completed, the unbiased estimator of $W_v = |R_{G'}(S \cup \{v\})| = w(R_{G'^*}(S^* \cup \{v^*\}))$ is given by

$$\hat{W}_v = \frac{T-1}{\sum_{1 \leq i \leq T} Y^i_{v^*}}.$$

The idea is based on the property of exponential distribution that ensures $Y^i_{v^*}$ follow an exponential distribution with the probability density function $e^{-\lambda_{v^*} x} \sum_{u^* \in R_{G'^*}(S^* \cup \{v^*\})} w(u^*)$. Cohen provides the following concentration result on the estimate.

THEOREM 1 (COHEN [1]). *For every vertex v in graph G',*

$$\textit{for } 0 < \epsilon < 1 \text{ , } \mathrm{Prob}[|\hat{W}_v - W_v| \geq \epsilon W_v] = exp(-\Omega(\epsilon^2 T)),$$
$$\textit{for } \epsilon \geq 1 \text{ , } \mathrm{Prob}[|\hat{W}_v - W_v| \geq \epsilon W_v] = exp(-\Omega(\epsilon T)).$$

We incorporate Cohen's algorithm into our greedy algorithm such that for each of the R generated graphs G' in each of the k rounds, Cohen's 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 MixedGreedyWC algorithm where the first round uses NewGreedyWC

Algorithm 3 NewGreedyWC(G, k)

```
initialize S = ∅, 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* ∈ V*
        for ℓ = 1 to T do
            for each v* ∈ V*, s^ℓ_{v*} = 0
            for each v* ∈ V*, generate random value X^ℓ_{v*} from
            the exponential distribution with mean 1/w(v*)
            for each v* ∈ V*, compute Y^ℓ_{v*} =
            min_{u*∈R_{G'*}(S*∪{v*})} X^ℓ_{u*}
            for each v* ∈ V*, s^ℓ_{v*} += Y^ℓ_{v*}
        end for
        for each v ∈ V \ S, s_v += (T − 1)/s^ℓ_{v*}
    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
```

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.

3. DEGREE DISCOUNT HEURISTICS

Even with the improved greedy algorithms we presented in Section 2, their running time is still large and may not be suitable for large social network graphs. A possible alternative is to use heuristics. In sociology literature, degree and other centrality-based heuristics are commonly used to estimate the influence of nodes in social networks [10].

Degree is frequently used for selecting seeds in influence maximization. Experimental results in [5] showed that selecting vertices with maximum degrees as seeds results in larger influence spread than other heuristics, but is still not as large as the influence spread produced by the greedy algorithms.

In this section, we propose *degree discount* heuristics, which nearly match the performance of the greedy algorithms for the IC model, while also improve upon the pure degree heuristic in other cascade models.

The general idea is as follows. let v be a neighbor of vertex u. If u has been selected as a seed, then when considering selecting v as a new seed based on its degree, we should not count the edge $\overline{vu}$ towards its degree. Thus we should discount v's degree by one due to the presence of u in the seed set, and we do the same discount on v's degree for every neighbor of v that is already in the seed set. This is a basic degree discount heuristic applicable to all cascade models, and is referred to as SingleDiscount in our experiment section.

For the IC model with a small propagation probability p, we derive a more accurate degree discount heuristic.[3] Since v is a neighbor of u that has been selected into the seed set, with probability at least p, v will be influenced by u, in which case we do not need to select v into the seed set. This is the reason why further discount is more accurate. When p is small, we may ignore indirect influence of v to multi-hop neighbors and focus on the direct influence of v to its immediate neighbors, which makes degree discount calculation manageable. This forms the guideline for us to compute the degree discount amount.

Let $N(v) = \{v\} \cup \{w \in V \mid \overline{vw} \in E\}$, and call it the *neighborhood* of v. Let $Star(v)$ be the subgraph with $N(v)$ as the vertices and edges incident to v as the edges. We compute the additional influence that v could make in the $Star(v)$ subgraph in order to derive a degree discount amount. Let t_v be the number of neighbors of vertex v that are already selected as seeds.

THEOREM 2. *In the IC model with propagation probability p, suppose that $d_v = O(1/p)$ and $t_v = o(1/p)$ for a vertex v.* [4] *The expected number of additional vertices in $Star(v)$ influenced by selecting v into the seed set is:*

$$1 + (d_v - 2t_v - (d_v - t_v)t_v p + o(t_v)) \cdot p. \tag{1}$$

PROOF. Let S_v be the set of t_v neighbors of v that have been selected as seeds. The probability that v is influenced by its immediate neighbors is $1 - (1-p)^{t_v}$. In this case, selecting v as a seed does not contribute additional influence in the graph.

If v is not influenced by any of the already selected seeds, which occurs with probability $(1-p)^{t_v}$, then the additional vertices in $Star(v)$ influenced by selecting v into the seed set include: (a) v itself with probability 1; and (b) each u in the remaining $d_v - t_v$ neighbors with probability p. Thus the additional vertices in $Star(v)$ influenced by v is $1 + (d_v - t_v) \cdot p$. Hence the overall expected number of additional vertices in $Star(v)$ influenced by v is

$$\begin{aligned}
&(1-p)^{t_v} \cdot (1 + (d_v - t_v) \cdot p) \\
&= (1 - t_v p + o(t_v p)) \cdot (1 + (d_v - t_v) \cdot p) \\
&\qquad \{\text{since } t_v p = o(1)\} \\
&= 1 + (d_v - 2t_v)p - (d_v - t_v)t_v p^2 + o(t_v p) \\
&\qquad \{\text{since } (d_v - t_v)p = O(d_v p) = O(1)\} \\
&= 1 + (d_v - 2t_v - (d_v - t_v)t_v p + o(t_v))p.
\end{aligned}$$

□

With the above theorem, we can compare it with the case when v has no neighbors selected yet as seeds (i.e., $t_v = 0$). In the latter case, the expected number of influenced vertices in $Star(v)$ is $1 + d_v \cdot p$. Comparing this with Equation (1), we conclude that for a vertex v with t_v neighbors already selected as seeds, we should discount v's degree by $2t_v + (d_v - t_v)t_v p$. For example, for a node v with $d_v = 200$, $t_v = 1$, and $p = 0.01$ (parameters similar to our experimental graphs), we should discount v's degree to about 196.

Although our calculation is only based on the $Star(v)$ graph and we do not consider other factors, such as indirect influence effects and selected seeds affecting the neighbors of v, we believe that the difference of those effects between the case $t_v = 0$ and $t_v > 0$ is negligible for small p. Our experimental results demonstrate that the degree discount based on Equation (1) matches very closely in influence spread to the best greedy algorithm.

The assumptions $d_v = O(1/p)$ and $t_v = o(1/p)$ are satisfied by our experimental settings and we believe are reasonable for other social networks. If we further have $d_v = o(1/p)$, we can show that the discount should be 2 even when we consider multi-hop neighborhoods as well as other influence effects. However, in the

[3] For the IC model with relatively large propagation probability p, the influence spread is not very sensitive to different algorithms and heuristics, because a giant connected component exists even after removing every edge with probability $1-p$. This has been reported in [5] with $p = 0.1$.

[4] We assume that p is a variable tending to 0, and other quantities such as $d_{\max}$ and t_v are functions of p.

Algorithm 4 DegreeDiscountIC(G, k)

```
initialize S = ∅
for each vertex v do
    compute its degree d_v
    dd_v = d_v
    initialize t_v to 0
end for
for i = 1 to k do
    select u = arg max_v{dd_v | v ∈ V \ S}
    S = S ∪ {u}
    for each neighbor v of u and v ∈ V \ S do
        t_v = t_v + 1
        dd_v = d_v − 2t_v − (d_v − t_v)t_v p
    end for
end for
output S
```

real-life graphs we considered in our experiments, d_v is typically larger than 100 while $p = 0.01$. Therefore, we cannot ignore the term $(d_v - t_v)t_v p$ in the discount.

Algorithm 4 implements the degree discount heuristic. Using Fibonacci heap, the running time of Algorithm 4 is $O(k \log n + m)$. Therefore, in theory we can already see that our degree discount heuristic is much faster than the original greedy algorithm with $O(knNm)$ running time or our improvement with $O(kNm)$ running time.

4. EXPERIMENTS

We conduct experiments for various algorithms on two real-life networks. The goal of our experiments is to show that we further improve the running time of the greedy algorithms with matching influence spreads, while our heuristic algorithms are orders of magnitude faster than all greedy algorithms with influence spread still close to those of the greedy algorithms.

4.1 Experiment setup

We extract two read-life academic collaboration networks from the paper lists in two different sections of the e-print arXiv[5], which is the same source used in the experimental study in [5]. Each node in the network represents an author, and the number of edges between a pair of nodes is equal to the number of papers the two authors collaborated. The first network is from the "High Energy Physics - Theory" section with papers form 1991 to 2003 and is denoted as NetHEPT, which contains $n_1 = 15,233$ nodes and $m_1 = 58,891$ edges. The second network is from the full paper list of the "Physics" section, denoted as NetPHY, which contains $n_2 = 37,154$ nodes and $m_2 = 231,584$ edges. The two graphs are available for download on the web at http://research.microsoft.com/en-us/people/weic/graphdata.zip. The experiments are run on a server with 2.33GHz Quad-Core Intel Xeon E5410 and 32G memory.

The two basic influence cascade models reported are the IC model and the WC model, as explained in Section 2. In the IC model, we mainly report results on a small propagation probability of $p = 0.01$, the value used in the experiments of [5], but also consider other values $p = 0.02$ and 0.05. Larger p values such as $p = 0.1$ are not considered due to its insensitivity to different algorithms (see Footnote 3).

We run the following set of algorithms under both IC and WC models on both networks.

[5]http://www.arXiv.org

- CELFGreedy: This is the original greedy algorithm (Algorithm 1) with the CELF optimization of [7]. The results on the original greedy algorithm are not reported since its influence is the same as the CELF optimization while its running time is too slow. We take $R = 20000$ to obtain accurate estimates.
- NewGreedyIC: The new greedy algorithm proposed for the IC model (Algorithm 2), with $R = 20000$.
- MixedGreedyIC: The mixed greedy algorithm for the IC model, in which the first round of the algorithm uses NewGreedyIC and the rest rounds use CELFGreedy.
- NewGreedyWC: The new greedy algorithm proposed for the WC model (Algorithm 3), with $R = 20000$ and $T = 5$.
- MixedGreedyWC: The mixed greedy algorithm for the WC model, in which the first round of the algorithm uses NewGreedyWC with $R = 20000$ and $T = 5$, and the rest rounds use CELFGreedy.
- DegreeDiscountIC: The degree discount heuristic for the IC model (Algorithm 4).
- SingleDiscount: A simple degree discount heuristic where each neighbor of a newly selected seed discounts its degree by one. This can be applied to all influence cascade models.
- Degree: As a comparison, a simple heuristic that selects the k vertices with the largest degrees, which is also evaluated in [5].
- Distance: As a comparison, a simple heuristic that selects the k vertices with the smallest average shortest-path distances to all other vertices, which is also evaluated in [5]. The distance of two disconnected vertices is set to n, the number of vertices in the graph.
- Random: As a baseline comparison, simply select k random vertices in the graph, which is also evaluated in [5].

To obtain the influence spread of the heuristic algorithms, for each seed set, we run the simulation of the IC or WC model on the networks 20000 times and take the average of the influence spread. This matches the accuracy of the greedy algorithms.

For all these algorithms, we compare their influence spreads with different seed set size ranging from 1 to 50. We also compare their running time for selecting $k = 50$ seeds for influence maximization.

To further evaluate the robustness of our proposed seed selection algorithms under different cascade models, we run simulations using the seeds selected in our algorithms on another model called the *linear threshold model* [5].

4.2 Experiment results

We summarize our experiment results for different influence cascade models and discuss their implications.

Independent cascade model. Figure 1 and Figure 2 show the influence spreads of various algorithms on the two collaboration graphs in the IC model with $p = 0.01$, which are very consistent. For ease of reading, in all influence spread figures, the legend ranks the algorithms top-down in the same order as the influence spreads of the algorithms when $k = 50$. All percentage difference reported below on influence spreads are for the case of $k = 50$.

The influence spreads of Random, Distance and Degree are in line with the results in [5]: Random as the baseline performs very badly, and simple Distance and Degree heuristics are better but are still significantly worse than the best algorithms such as CELFGreedy (Distance is 20.9% and 46.3% lower, and Degree is 8.7% and 16.3% lower, on NetHEPT and NetPHY, respectively). SingleDiscount heuristic, although just a simple adjustment to the Degree heuristic, reduced approximately half of the

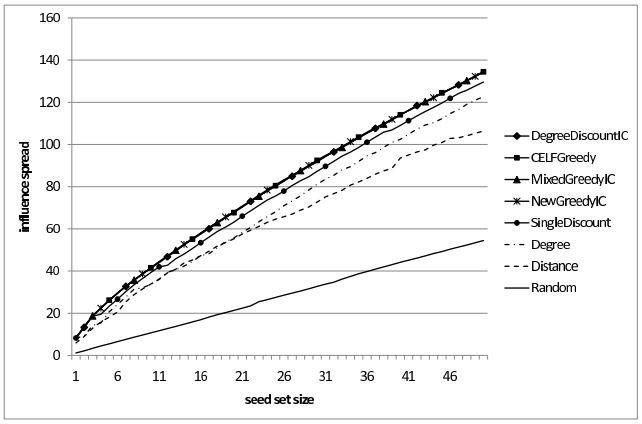

Figure 1: Influence spreads of different algorithms on the collaboration graph NetHEPT under the independent cascade model ($n = 15,233$, $m = 58,891$, and $p = 0.01$).

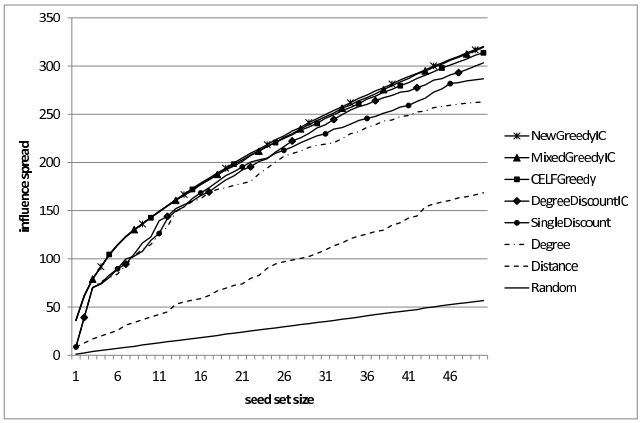

Figure 2: Influence spreads of different algorithms on the collaboration graph NetPHY under the independent cascade model ($n = 37,154$, $m = 231,584$, and $p = 0.01$).

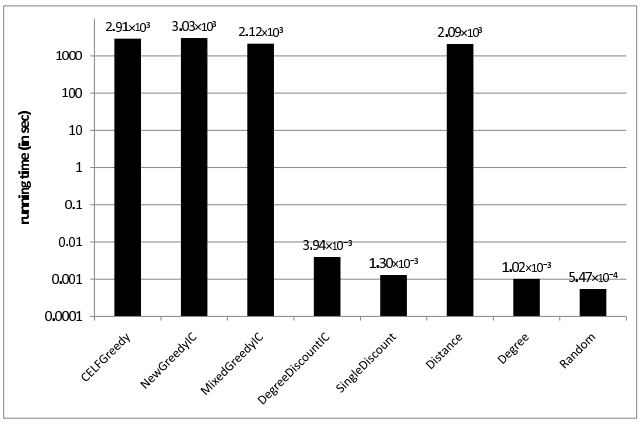

Figure 3: Running times of different algorithms on the collaboration graph NetHEPT under the independent cascade model ($n = 15,233$, $m = 58,891$, $p = 0.01$, and $k = 50$).

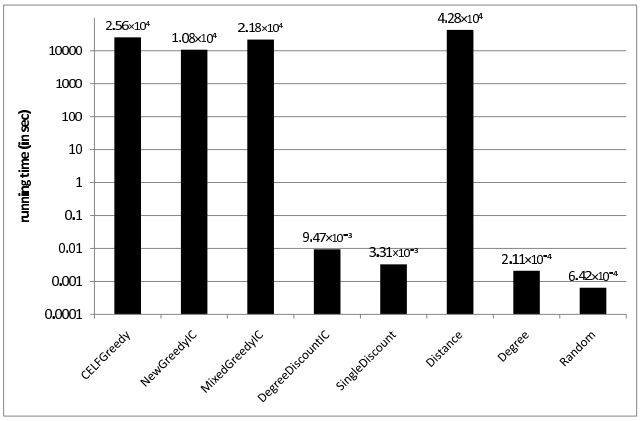

Figure 4: Running times of different algorithms on the collaboration graph NetPHY under the independent cascade model ($n = 37,154$, $m = 231,584$, $p = 0.01$, and $k = 50$).

gap between Greedy and Degree (3.6% and 8.6% lower than the greedy algorithm, on NetHEPT and NetPHY, respectively). Our NewGreedyIC and MixedGreedyIC essentially matches CELFGreedy on both graphs. The most interesting result is on our DegreeDiscountIC heuristic. It performs extremely well, essentially matching the influence spread of the CELFGreedy algorithm in NetHEP, while in NetPHY, it is also very close to CELFGreedy (only 3.4% lower for $k = 50$).

Figure 3 and Figure 4 report the running times of different algorithms for selecting $k = 50$ seeds in the two graphs. All results are measured on reasonably efficient implementation of the various algorithms. Notice that the y-axis is in log scale. The running time of our NewGreedyIC is 57.9% lower than CELFGreedy in NetPHY but is 4.2% higher than CELFGreedy in NetHEPT. This indicates that the benefit of our improvement verses CELF improvement may depend on the actual graph structure. Our MixedGreedyIC, however, combines the benefit of the two improvement and is always better than CELFGreedy: it saves 27% and 15% running times of CELFGreedy in NetHEPT and NetPHY, respectively. The new heuristics DegreeDiscountIC and SingleDiscount run extremely fast, only take a few milliseconds to finish, and are more than six orders of magnitude faster than all greedy algorithms. The Distance heuristic has running time in the range of the greedy algorithm while its influence spread is much worse than the greedy algorithm, indicating that it is not a suitable choice for the influence maximization problem.

We also run simulations with $p = 0.02$ and $p = 0.05$. For influence spread, other than that its absolute scale multiplies, the general trend and relative performance are similar to the case of $p = 0.01$. More interestingly is the running time of different algorithms. The running time of our NewGreedyIC and DegreeDiscountIC algorithms are not affected much by different p values, e.g. on NetHEPT NewGreedyIC always takes around 3000 seconds and DegreeDiscountIC always takes about 4 milliseconds. This is because both algorithms spend major portion of time in graph traversal to determine edge removals, the running time of which is independent of p. However, CELFGreedy becomes slower and slower as p increases, e.g. on NetHEPT CELFGreedy takes 9.28×10^3 seconds for $p = 0.02$ and 1.11×10^5 seconds for $p = 0.05$. This means that CELF optimization is less effective when p increases.

Weighted cascade model. Figure 5 and Figure 6 show the influence spreads of various algorithms on the two collaboration graphs in the WC model. Our NewGreedyWC and MixedGreedyWC algorithms always match the CELFGreedy algorithm in both graphs. SingleDiscount again always performs better than Degree and Distance, and is fairly close to the greedy algorithm in NetHEPT (9.1%) but is not as well in NetPHY (28.7%). DiscountDegreeIC is a heuristic specifically tuned for the IC model and is not included here for the WC model.

Figure 7 and Figure 8 report the running times of different algorithms for selecting $k = 50$ seeds in the two graphs. As ex-

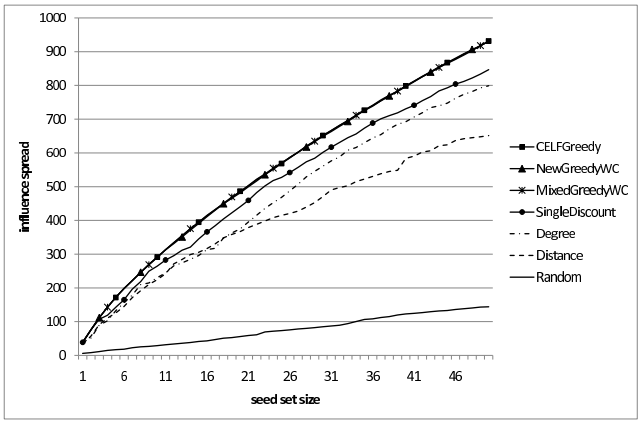

Figure 5: Influence spreads of different algorithms on the collaboration graph NetHEPT under the weighted cascade model ($n = 15,233$ and $m = 58,891$).

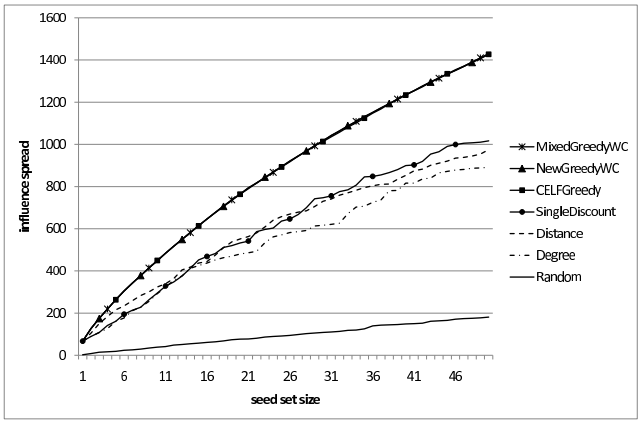

Figure 6: Influence spreads of different algorithms on the collaboration graph NetPHY under the weighted cascade model ($n = 37,154$ and $m = 231,584$).

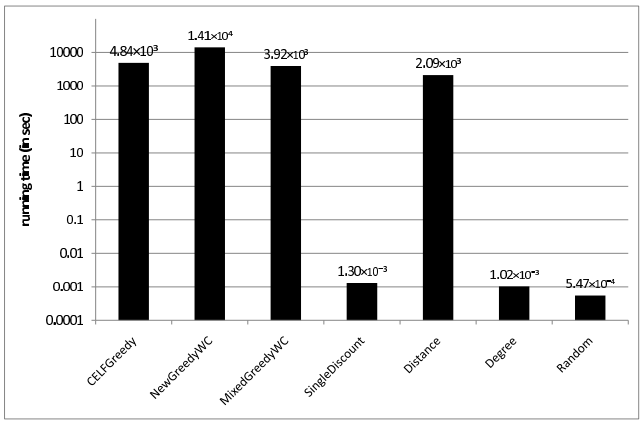

Figure 7: Running times of different algorithms on the collaboration graph NetHEPT under the weighted cascade model ($n = 15,233$, $m = 58,891$, and $k = 50$).

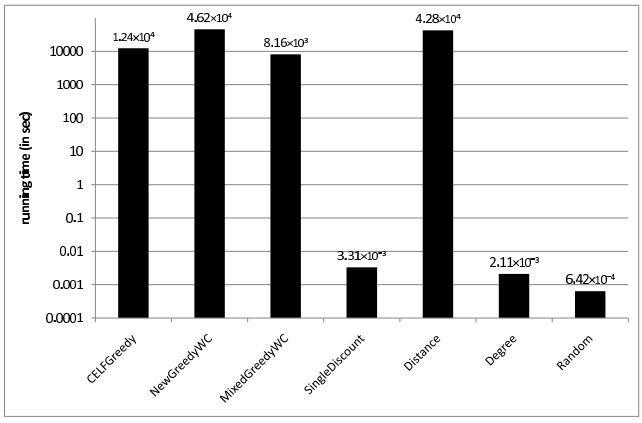

Figure 8: Running times of different algorithms on the collaboration graph NetPHY under the weighted cascade model ($n = 37,154$, $m = 231,584$, and $k = 50$).

pected, due to its multiple traversals of every generate graph, NewGreedyWC is slower than CELFGreedy. However, similar to the case in the IC model, MixedGreedyWC always runs faster than CELFGreedy, with 19.0% and 34.4% savings in running times in NetHEPT and NetPHY, respectively. SingleDiscount does not depend on the influence cascade model and thus have the same running time numbers as reported in the IC model — it is more than six orders of magnitude faster than all greedy algorithms.

Linear threshold model. In the linear threshold (LT) model, every vertex has a threshold uniformly and randomly chosen from 0 to 1, and a vertex is activated when the fraction of its activated neighbors reaches its threshold [5]. Threshold models have been proposed and studied in sociology as a reasonable model for influence in social networks (e.g. [4, 9]). Since our improved greedy algorithms and heuristics are based either on the IC model or WC model, a natural question to ask is whether they are still effective in the LT model. To verify this, we take the seeds selected by our algorithms, in particular the MixedGreedyWC, MixedGreedyIC, DegreeDiscountIC, and SingleDiscount algorithms ($p = 0.01$ for the IC algorithms), and run simulations in the LT model on both networks.

Figure 9 and Figure 10 report the results on influence spreads of our algorithms as well as other algorithms for comparison. The CELFGreedy algorithm is run on the LT model so it is not a surprise that it generates the best influence spread. In both graphs, MixedGreedyWC matches very well with CELFGreedy, suggesting that the dynamics of the WC model is similar to the LT model, as also mentioned in [5]. Our other greedy and heuristics perform reasonably well, especially in the NetHEPT graph. A surprise is that DegreeDiscountIC performs quite well in both graphs, and is even much better than MixedGreedyIC in the NetPHY graph. This suggests that appropriate degree discounts may be applicable to other influence cascade models and worth further study.

As for the running time, only CELFGreedy has different running time and all other algorithms have the same running time as reported earlier, since they are either independent of cascade models or based on the IC or WC model rather than the LT model. The running time of CELFGreedy is 9.03×10^3 seconds on NetHEPT graph and 4.47×10^4 seconds on the NetPHY graph, which are at about the same level as greedy algorithms in other cascade models.

4.3 Discussion on the results

Several conclusions can be drawn from the experimental results reported above. First, by combining the advantage of our improvement and CELF improvement to the greedy algorithm, our mixed algorithms further improve the running time of the CELF optimized algorithm (15% to 34%)[6] while matching the influence spread to the original algorithm. Second, and more importantly, the im-

[6]The numbers are based on the IC model with $p = 0.01$ and the WC model. For IC model with larger p's such as $p = 0.02$ and 0.05, our experiments show that we may achieve an order of magnitude speedup with NewGreedyIC or MixedGreedyIC.

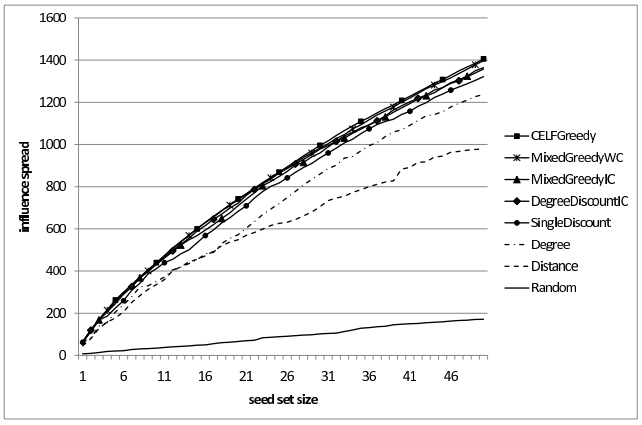

Figure 9: Influence spreads of different algorithms on the collaboration graph NetHEPT under the linear threshold model ($n = 15,233$ and $m = 58,891$).

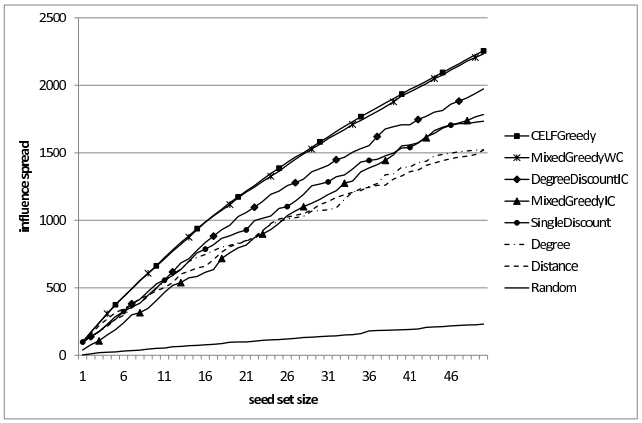

Figure 10: Influence spreads of different algorithms on the collaboration graph NetPHY under the linear threshold model ($n = 37,154$ and $m = 231,584$).

proved heuristics we propose are able to reduce the running time by more than six orders of magnitude (from hours to milliseconds) with little or mild compromise on the influence thread. In particular, our DegreeDiscountIC heuristic shows very impressive results in the independent cascade model: it only takes less than one-millionth of time of any greedy algorithm but exhibits no or less than 3.5% degrade in influence spread. The SingleDiscount heuristic also performs reasonably well, and is always better than the traditional Distance and Degree heuristics. Third, even though our new algorithms and heuristics are developed for the independent cascade model and the weighted cascade model, they are robust and can be applied to other influence models such as the linear threshold model.

Based on these results, we suggest the following treatment to the influence maximization problem. When short running time is important, we should choose SingleDiscount and DegreeDiscountIC. DegreeDiscountIC usually performs better in influence spread, while SingleDiscount is potentially applicable to a wider range of influence cascade models. When running time is not a major concern but guaranteed influence spread is essential, we can use MixedGreedyIC or MixedGreedyWC algorithms for seed selection.

5. CONCLUDING REMARKS

In this paper, we propose the efficient algorithms and heuristics for the influence maximization problem. We both reduce the running time of existing best greedy algorithms while maintaining the influence spread guarantee, and propose new heuristics that significantly improve the influence spread while running more than six orders of magnitude faster than all greedy algorithms. We believe that rather than trying to further reduce the running time of the greedy algorithms, we should focus our research effort in searching for more effective heuristics for different influence cascade models. With their influence spread getting close to that of the greedy algorithm and their extremely fast speed, they are likely to be the scalable solutions to the influence maximization problem for large-scale real-life social networks.

There are several future directions for this research. First, our current degree discount heuristic is derived from the independent cascade model. We plan to look into appropriate degree discount strategies for other cascade models. Second, the current influence maximization problem is simplified, without considering other features in the social networks. We plan to investigate how to extract community structures and how to utilize these community structures to facilitate influence maximization in social networks.

6. REFERENCES

[1] E. Cohen. Size-estimation framework with applications to transitive closure and reachability. *J. Comput. Syst. Sci.*, 55(3):441–453, 1997.

[2] D. Coppersmith and S. Winograd. Matrix multiplication via arithmetic progressions. *J. Symb. Comput.*, 9(3):251–280, 1990.

[3] P. Domingos and M. Richardson. Mining the network value of customers. In *Proceedings of the 7th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 57–66, 2001.

[4] M. Granovetter. Threshold models of collective behavior. *American J. of Sociology*, 83(6):1420–1443, 1978.

[5] D. Kempe, J. M. Kleinberg, and É. 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 137–146, 2003.

[6] 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 Discovery in Databases*, pages 259–271, 2006.

[7] 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 420–429, 2007.

[8] M. Richardson and P. Domingos. Mining knowledge-sharing sites for viral marketing. In *Proceedings of the 8th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 61–70, 2002.

[9] T. C. Schelling. *Micromotives and Macrobehavior*. Norton, 1978.

[10] S. Wasserman and K. Faust. *Social Network Analysis: Methods and Applications*. Cambridge University Press, 1994.