Top-K Influential Nodes in Social Networks: A Game Perspective

Yu Zhang
Key Laboratory of Machine Perception (MOE) &
Dept. of Computer Science, Peking University
Beijing, China
yuz9@illinois.edu

Yan Zhang
Key Laboratory of Machine Perception (MOE) &
Dept. of Machine Intelligence, Peking University
Beijing, China
zhy@cis.pku.edu.cn

ABSTRACT

Influence maximization, the fundamental of viral marketing, aims to find top-K seed nodes maximizing influence spread under certain spreading models. In this paper, we study influence maximization from a game perspective. We propose a Coordination Game model, in which every individuals make their decisions based on the benefit of coordination with their network neighbors, to study information propagation. Our model serves as the generalization of some existing models, such as Majority Vote model and Linear Threshold model. Under the generalized model, we study the hardness of influence maximization and the approximation guarantee of the greedy algorithm. We also combine several strategies to accelerate the algorithm. Experimental results show that after the acceleration, our algorithm significantly outperforms other heuristics, and it is three orders of magnitude faster than the original greedy method.

CCS CONCEPTS

• **Information systems → Data mining**; • **Theory of computation → Design and analysis of algorithms**;

KEYWORDS

influence maximization; coordination game model; social networks; viral marketing

1 INTRODUCTION

Social networks play an important role in information diffusion. They give us motivation to use a small subset of influential individuals in a social network to activate a large number of people. Kempe et al. [9] build a theoretical framework of influence maximization, aiming to find top-K influential nodes under certain spreading models. They discuss two popular models - Independent Cascade (IC) model and Linear Threshold (LT) model and propose a greedy algorithm with $(1 - 1/e - \epsilon)$-approximation rate.

Easley and Kleinberg [7] divide the cause of information propagation into two categories: information effects and direct-benefit effects. Obviously, IC model and LT model belong to the former one, while we focus on the latter one. In most spreading models, each node has two states: active and inactive. Equivalently saying, it has two choices. In our Coordination Game (CG) model, we regard information diffusion as the process of individual decision-making. As individuals make their decisions based on the benefit of coordination with their network neighbors, a particular pattern of behavior can begin to spread across the links of the network.

Influence maximization under CG model is useful in viral marketing. Let us recall the example in [9]. A company would like to market a new product, hoping it will be adopted by a large fraction of the network. The company can initially target a few influential nodes by giving them free samples of the product. Then other nodes will probably switch to using the new product because of the following two reasons: (1) They have higher evaluation of the new product than the old one. (2) They have to coordinate with their neighbors because using different products may reduce their benefits. (e.g., people using different operating systems may have compatibility problems when working together, and users from different kinds of social media platforms cannot communicate with each other timely.) Our model describes these two reasons precisely.

In this paper, we study how to find Top-K influential nodes under CG model. We first propose our model which serves as the generalization of some well-known spreading models, such as Majority Vote model [2] and Linear Threshold model [9]. We then prove some theoretical results under CG model, including NP-hardness of the optimization problem itself and #P-hardness of computing the objective function. Then we try to find a good approximation algorithm for the problem. We embed our CG model into the scenario of *general diffusion process* [12], and prove that the objective function is monotone and submodular if and only if the *cumulative distribution function* of people's threshold is concave, in which case the greedy algorithm can return a $(1-1/e-\epsilon)$ approximate solution.

As a traditional method, Kempe et al. [9] use 10,000 times of Monte Carlo simulations to approximate the objective function, but it costs too much time on large-scale networks. To accelerate our algorithm, we use two efficient heuristics - LazyForward [10] and StaticGreedy [6]. Experimental results show that our `Greedy` and `Greedy++` algorithms can activate more nodes than other heuristics. Moreover, `Greedy++` runs faster than `Greedy` by three orders of magnitude.

Related Work. Kempe et al. [9] first build an algorithmic framework of influence maximization by transforming it into a discrete optimization problem. After their work, a lot of efforts have been made on efficient computing methods of the objective function. Some methods aim to reduce the number of trials that need Monte Carlo simulations, such as `CELF` [10]. Other researchers focus on how to calculate the influence spread efficiently. For instance, Chen et al. [3, 4] use arborescences or DAGs to represent the original

SIGIR'17, August 07-11, 2017, Shinjuku, Tokyo, Japan

ACM ISBN 123-4567-24-567/08/06…$15.00.
https://doi.org/10.475/123_4

		v: A	v: B
u	A	p_{uA}, p_{uB}	0, 0
	B	0, 0	p_{vA}, p_{vB}

Figure 1: Payoff matrix of the coordination game.

graph. Cheng et al. propose a StaticGreedy strategy [6] and a self-consistent ranking method [5].

Morris [11] is the first to propose a coordination game model in contagion. This model is also discussed detailedly in Easley and Kleinberg's textbook [7]. We will extend this model by introducing some random factors into utility values.

2 MODEL

In a social network $G = (V, E)$, we study a situation in which each node has a choice between two behaviors, labeled A and B. If nodes u and v are linked by an edge, then there is an incentive for them to have their behaviors match. We use a game model to describe this situation. There is a coordination game on each edge $(u, v) \in E$, in which players u and v both have two strategies A and B. The payoffs are defined as follows:

(1) if u and v both adopt strategy A, they will get payoffs $p_{uA} > 0$ and $p_{vA} > 0$ respectively;

(2) if they both adopt strategy B, they will get payoffs $p_{uB} > 0$ and $p_{vB} > 0$ respectively;

(3) if they adopt different strategies, they each get a payoff of 0.

The payoff matrix is shown in Figure 1.

We define the total payoff of player u as the sum of the payoffs it gets from all coordination games with its neighbors $N(u) = \{v|(u, v) \in E\}$. If u can get a higher total payoff when it adopt A than that when it adopt B, it will choose strategy A. Otherwise it will choose strategy B.

According to the actual situation, we have the following assumptions for the payoffs:

(1) All the p_{uA} and p_{uB} $(u \in V)$ may not be equal to each other because each person in the social network values behaviors A and B differently.

(2) p_{uA} and p_{uB} $(u \in V)$ can either be constants or independent and identically distributed random variables because the cascading behaviors in networks are always considered to have determinate principles with some stochastic factors.

Suppose u knows all the choices of its neighbors: there are x_B nodes adopting B and $x_A = deg(u) - x_B$ nodes adopting A. Obviously, u will adopt B if and only if

$$p_{uB}x_B \geq p_{uA}x_A = p_{uA}(deg(u) - x_B), \tag{1}$$

or

$$x_B \geq \frac{p_{uA}}{p_{uA} + p_{uB}} deg(u) = \delta_u deg(u), \quad \delta_u \in [0, 1]. \tag{2}$$

Influence Maximization Problem. Suppose now the market is dominated by A (i.e., all of the nodes in the network choose A). Given a constant k, we want to find a seed set $S_0 \subseteq V$, $|S_0| \leq k$. Initially we let each node in S_0 adopt B (and they will never change their choices again). Time then runs forward in unit steps. In each step, each node decides whether to switch from strategy A to strategy B according to the payoff-maximization principle. We can regard the evolution of nodes' choices as a spreading process of B in the network. The spread of behavior B will finally stop in at most $n = |V|$ steps.

We define $S_i = |\{u \in V | u \text{ adopts } B \text{ in step } i\}|$ $(i = 1, 2, ..., n)$. Our objective function is (the expectation of) the nodes affected by B at last, or

$$\sigma(S_0) = \mathbb{E}_{\{p_{uA}, p_{uB}|u \in V\}}[|S_n|] = \mathbb{E}_{\{\delta_u|u \in V\}}[|S_n|]. \tag{3}$$

Our purpose is to maximize $\sigma(S_0)$ subject to $|S_0| \leq k$.

The CG model can be regarded as the generalization of the following two well-known spreading models.

Majority Vote Model. Suppose all the p_{uA} $(u \in V)$ are constants and are equal to each other. So are all the p_{uB} $(u \in V)$. Equivalently, let

$$p_A = p_{uA}, \ \ p_B = p_{uB}, \ \ \delta = \delta_u = \frac{p_A}{p_A + p_B}, \ \ \forall u \in V. \tag{4}$$

δ is a constant threshold same to every nodes. When $p_A = p_B$, or $\delta = \frac{1}{2}$, the spreading model is called Majority Vote model, which is extensively studied in [2].

Linear Threshold Model. If we set $p_{uA} = 1$ and let p_{uB} follow a continuous power-law distribution, i.e., the *probabilistic density function* of p_{uB} is

$$f_B(x) = \frac{\alpha}{(x+1)^\gamma}, \ x \geq 0, \ \gamma > 1, \ \alpha = \frac{1}{\int_0^\infty \frac{1}{(x+1)^\gamma} \mathrm{d}x} = \gamma - 1, \tag{5}$$

then $\forall 0 \leq x \leq 1$,

$$\begin{aligned} \Pr[\delta_u \leq x] &= \Pr[\frac{1}{1 + p_{uB}} \leq x] = \Pr[p_{uB} \geq 1/x - 1] \\ &= \int_{1/x-1}^{+\infty} f_B(t)\mathrm{d}t = -(t+1)^{-\gamma+1}\Big|_{1/x-1}^{+\infty} = x^{\gamma-1}. \end{aligned} \tag{6}$$

If $\gamma = 2$, we will have $\delta_u \sim U[0, 1]$. This is the famous Linear Threshold model where the weight on each edge adjacent to node u is $1/deg(u)$ (i.e., $b_{vu} = \frac{1}{deg(u)}, \forall u, v \in V$).

Hardness. Under CG model, we have the following hardness result.

Theorem 2.1. *(1) Influence maximization under CG model is NP-hard. (2) Computing the objective function under CG model is #P-hard.*

Proof. (1) Chen [2] proves the NP-hardness of Influence Maximization under Majority Vote model with $\delta = \frac{1}{2}$, which is enough to demonstrate the first result.

(2) Chen et al. [4] prove it is #P-hard to compute exact influence in general networks under LT model. They use the settings that $b_{vu} = const, \forall u, v \in V$ in their proof. We modify the proof and get the hardness result under our settings.[1] We reduce this problem from the problem of counting simple paths in a directed graph. Given a directed graph $G = (V, E)$, counting the total number of simple paths in G is #P-hard [14]. Let $n = |V|$ and $D = \max_{v \in V} deg_{in}(v)$. From G, we construct $n + 1$ graphs $G_1, G_2, ..., G_{n+1}$. To get G_i $(1 \leq i \leq n + 1)$, we first add $D + i - deg_{in}(v)$ "branching nodes" linking to node v for all $v \in V$. And then we add a node s linking

[1] Note that $b_{vu} = const$ is not a special case of CG model.

to all nodes in V. Thus each node in G_i has $D+i+1$ in-links except "branching nodes" and s.

According to our assumption, the weight on each edge in G_i is $w_i = \frac{1}{D+i+1}$. Let $S_0 = \{s\}$ and $\mathcal{P}$ denote the set of all simple paths starting from s in G_i. (Note that $\mathcal{P}$ is identical in all G_i because "branching nodes" are unreachable from s.) According to [4], we have

$$\sigma_{G_i}(S_0) = \sum_{\pi \in \mathcal{P}} \prod_{e \in \pi} w_i, \quad (1 \le i \le n+1), \tag{7}$$

where $\sigma_{G_i}(S_0)$ means $\sigma(S_0)$ in G_i. Let B_j be the set of simple paths of length j in $\mathcal{P}$ $(0 \le j \le n)$. We have

$$\sigma_{G_i}(S_0) = \sum_{j=0}^{n} \sum_{\pi \in B_j} \prod_{e \in \pi} w_i = \sum_{j=0}^{n} \sum_{\pi \in B_j} w_i^j = \sum_{j=0}^{n} w_i^j |B_j|. \tag{8}$$

We want to solve these $n+1$ linear equations with $n+1$ variables $|B_0|, |B_1|, ..., |B_n|$. Since the coefficient matrix is a Vandermonde matrix, $(|B_0|, |B_1|, ..., |B_n|)$ is unique and easy to compute.

Finally, we notice that for each $j = 1, 2, ..., n$, there is a one-to-one correspondence between paths in B_j and simple paths of length $j-1$ in G. Therefore, $\sum_{j=1}^{n} |B_j|$ is the total number of simple paths in G. We complete our reduction. □

3 ALGORITHMS

Submodularity. To find a greedy algorithm with approximation guarantee, the submodularity of the objective function is necessary. We first recall the general diffusion process defined by Mossel and Roch in [12].

Suppose each node v in the social network $G = (V, E)$ has a threshold $\theta_v \sim U[0,1]$ *i.i.d* and a "local" spreading function $f_v : 2^V \to [0,1]$. Initially there is a seed set $S_0 \subseteq V$. In each step $t \ge 1$,

$$S_t = S_{t-1} \cup \{v | v \in V - S_{t-1} \ \wedge \ f_v(S_{t-1}) \ge \theta_v\}. \tag{9}$$

The spreading process will stop in at most $n = |V|$ steps. So the objective function is $\sigma(S_0) = \mathbb{E}_{\{\theta_u | u \in V\}}[|S_n|]$.

We can embed our model into the scenario of the general diffusion process.

Let F_δ be the cumulative distribution function of δ_u. Since $\delta_u \in [0,1]$, we have $F_\delta(0) = 0$ and $F_\delta(1) = 1$. $\forall v$ and S, let

$$\theta_v = F_\delta(\delta_v) \ \text{ and } \ f_v(S) = F_\delta(\frac{|S \cap N(v)|}{deg(v)}). \tag{10}$$

Suppose F_δ is continuous and strictly monotone increasing in $[0,1]$, then F_δ^{-1} exists, and $\forall x \in [0,1]$,

$$\Pr[F_\delta(\delta_v) \le x] = \Pr[\delta_v \le F_\delta^{-1}(x)] = F_\delta(F_\delta^{-1}(x)) = x. \tag{11}$$

So $F_\delta(\delta_v) \sim U[0,1]$. Therefore

$$\begin{aligned} f_v(S) \ge \theta_v &\iff F_\delta(\frac{|S \cap N(v)|}{deg(v)}) \ge \theta_v \\ &\iff |S \cap N(v)| \ge F_\delta^{-1}(\theta_v) deg(v) \\ &\iff |S \cap N(v)| \ge \delta_v deg(v). \end{aligned} \tag{12}$$

Lemma 3.1. *Suppose F_δ is continuous and strictly monotone increasing in $[0,1]$, f_v is monotone and submodular for any node v (in any graph) **iff** F_δ is concave in $[0,1]$.*

Proof. ($\Leftarrow$) If F_δ is concave in $[0,1]$, let $g_v(S) = \frac{|S \cap N(v)|}{deg(v)}$, which is a modular function. It is easy to prove that the composition of a concave function and a modular function is submodular. Therefore $f_v = F_\delta \circ g_v$ is also monotone and submodular.

($\Rightarrow$) If F_δ is not concave in $[0,1]$, then $\exists a, b, \lambda \in [0,1]$ such that

$$\lambda F_\delta(a) + (1-\lambda) F_\delta(b) > F_\delta(\lambda a + (1-\lambda) b). \tag{13}$$

Since F_δ is (uniformly) continuous and bounded, if we pick up three rational numbers $\frac{N_1}{M}$, $\frac{N_2}{M}$ and $\frac{p}{q}$ which are very close to a, b, λ respectively, we will have

$$\frac{p}{q} F_\delta(\frac{N_1}{M}) + \frac{q-p}{q} F_\delta(\frac{N_2}{M}) > F_\delta(\frac{N_1 p + N_2(q-p)}{Mq}) = F_\delta(\frac{N_3}{Mq}). \tag{14}$$

Let $X_i = (\frac{i}{Mq}, F_\delta(\frac{i}{Mq}))$ be the points on the curve of F_δ $(i = N_1 q, ..., N_2 q)$ and l_0 be the line across $X_{N_1 q}$ and $X_{N_2 q}$. We know that X_{N_3} is below l_0. Therefore $\exists K_1 \le N_3 - 1$ and $K_2 \ge N_3$ such that

(1) X_{K_1} is above or in l_0 while X_{K_1+1} is below l_0.

(2) X_{K_2} is below l_0 while X_{K_1+1} is above or in l_0.

Let l_1 be the line across X_{K_1} and X_{K_1+1} and let l_2 be the line across X_{K_2} and X_{K_2+1}. We know that $k(l_1) < k(l_0) < k(l_2)$, where $k()$ is the slope of the line.

Assume there is a node v with Mq neighbors. Let S be the set of v's K_1 neighbors and T be the set of v's K_2 neighbors, where $S \subset T$. There is another neighbor $u \notin T$. Therefore

$$\begin{aligned} &f_v(T \cup \{u\}) - f_v(T) = F_\delta(\frac{K_2+1}{Mq}) - F_\delta(\frac{K_2}{Mq}) = \frac{k(l_2)}{Mq} \\ &> \frac{k(l_1)}{Mq} = F_\delta(\frac{K_1+1}{Mq}) - F_\delta(\frac{K_1}{Mq}) = f_v(S \cup \{u\}) - f_v(S), \end{aligned} \tag{15}$$

which violates the submodularity of f_v. □

It is not difficult for us to understand Lemma 1 intuitively because submodularity can be considered as a kind of concavity. F_δ being concave in $[0,1]$ means that the distribution of people's threshold has a negative skewness, or they tend to have a higher evaluation on new products than old ones. This assumption is reasonable in some cases (e.g., the mobile phone market). F_δ being continuous and strictly monotone increasing in $[0,1]$ is a technical assumption instead of an essential one. We define these two assumptions as the *concave threshold property*.

For the general diffusion process, Mossel and Roch [12] have proved that $\sigma(S_0)$ is monotone and submodular if and only if f_v is monotone and submodular for any $v \in V$. Using this result and Lemma 1, we can get Theorem 2 immediately.

Theorem 3.2. *$\sigma(S_0)$ is monotone and submodular **iff** F_δ satisfies the concave threshold property.*

Theorem 2 provides a strong tool to judge the objective function's submodularity under certain spreading models. For example, under Majority Vote model, $\sigma(S_0)$ is not submodular because $F_\delta(x) = \mathbb{I}(x \ge \delta)$ is not concave in $[0,1]$, where $\mathbb{I}(\cdot)$ is the indicator function. In contrast, under Linear Threshold model, $\sigma(S_0)$ is submodular because $F_\delta(x) = x$ is concave in $[0,1]$.

Up till now, we have proved the monotonicity and submodularity of the objective function under CG model with some necessary assumptions. Using the result in [9], the greedy algorithm given in Algorithm 1 (`Greedy`) returns a $(1 - 1/e - \epsilon)$-approximate solution.

Algorithm 1 Greedy(k, σ)

initialize $S_0 = \emptyset$
for $i = 1$ to k **do**
  select $u = argmax_{v \in V - S_0}(\sigma(S_0 \cup \{v\}) - \sigma(S_0))$
  $S_0 = S_0 \cup \{u\}$
end for
output S_0

Algorithm 2 Greedy++(k, σ, R')

initialize $S_0 = \emptyset$
for $i = 1$ to R' **do**
  generate the threshold δ_v ($\forall v \in V$) for snapshot G_i
end for
for all $v \in V$ **do**
  $\Delta_v = +\infty$ //initialize the marginal gain of each node
end for
for $i = 1$ to k **do**
  for all $v \in V - S_0$ **do**
    $cur_v = false$
  end for
  while *true* **do**
    $u = argmax_{v \in V - S_0} \Delta_v$ //maintain a priority queue
    if cur_u **then**
      $S_0 = S_0 \cup \{u\}$
      break
    else
      $\Delta_u = \frac{1}{R'} \sum_{i=1}^{R'} (\sigma_{G_i}(S_0 \cup \{u\}) - \sigma_{G_i}(S_0))$
      reinsert u into the priority queue and heapify
      $cur_u = true$
    end if
  end while
end for
output S_0

The algorithm simply selects seed nodes one by one, and each time it always selects the node that provides the largest marginal gain of the objective function.

Speeding-Up Algorithm. Due to the hardness of computing $\sigma(S_0)$, we use two efficient heuristics - LazyForward [10] and StaticGreedy [6] to accelerate our algorithm.

We maintain a priority queue. When finding the next node, we go through the nodes in decreasing order of their marginal gain. If the marginal gain of the top node has not been updated, we recompute it, and insert it into the priority queue again. The correctness of this lazy procedure can be guaranteed due to the submodularity of the objective function.

Instead of conducting a huge number of Monte Carlo simulations each time, we generate a rather small number of snapshots G_i ($i = 1, 2, ..., R'$) at the very beginning. In all the iterations, we run the simulation on these snapshots and use the average number of influenced nodes $\frac{1}{R'} \sum_{i=1}^{R'} \sigma_{G_i}(S_0)$ to estimate the objective function $\sigma(S_0)$.

We name the accelerated algorithm as Greedy++.

4 EXPERIMENTS

To test the effectiveness and efficiency of our Greedy and Greedy++ algorithms, we conduct experiments on three real-world networks and compare our algorithms with other existing heuristics.

Datasets. The three real-world datasets include two collaboration networks NetHEPT and NetPHY[2], and one online social network Epinions[3]. We summarize the statistical information of the these datasets in Table 1.

Table 1: Statistical information of three datasets.

Datasets	$\|V\|$	$\|E\|$	Type
NetHEPT	15,233	58,991	Undirected
NetPHY	37,154	231,584	Undirected
Epinions	75,879	508,837	Directed

Algorithms. A total of five algorithms are tested. Besides of Greedy and Greedy++ proposed in this paper, we use other three heuristic algorithms as benchmark methods.

(1) PageRank chooses nodes with the largest PageRank value. Since influential nodes are considered to have a large number of out-links, while nodes with high PageRank value are considered to have lots of in-links, we first change the direction of all edges in the graph and then run PageRank algorithm. We use $\alpha = 0.9$ as the random jump parameter. (2) Degree chooses nodes with the largest out-degree. (3) Random chooses nodes at random.

There are a lot of other efficient algorithms to solve the influence maximization problem under IC model or LT model, such as PMIA [3], LDAG [4], IMRank [5], and IMM [13]. But they cannot be applied in CG model directly, and we will not put them into the comparison.

Effectiveness. We first compare the effectiveness of Greedy and Greedy++ with other algorithms by showing influence spread (i.e., $|S_n|$) of the obtained seed set.

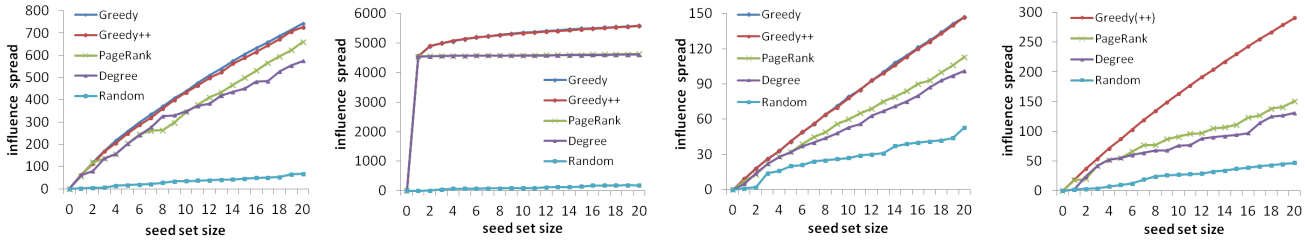
In our CG model, distribution of δ_u can be various. We run influence maximization algorithms under four different spreading models in NetHEPT. In the four models, we set δ_u to be X, X^2, $\sqrt{X}$ and 0.5 respectively, where $X \sim U[0, 1]$. Therefore the distribution function $F_\delta(x)$ is x, $\sqrt{x}$, x^2 and $\mathbb{I}(x \geq 0.5)$ corresponding to the four cases.

Figure 2 shows our experimental results. In Figure 2, Greedy++ consistently matches the performance of Greedy and significantly outperforms other heuristic algorithms in all cases. According to Theorem 2, the first two cases are submodular cases, while the other two are not. Therefore the approximation rates of Greedy and Greedy++ are not guaranteed in the third and the fourth cases. But our experimental results indicate that they still perform well in these cases. Besides, all the curves of Greedy and Greedy++ are concave no matter in submodular or nonsubmodular cases. In two larger graphs NetPHY and Epinions, we get similar experimental results, which are shown in Figures 3(a) and 3(b).

Note that PageRank and Degree perform worse in nonlinear threshold cases than they do in the linear case. The reason can be explained as follows:

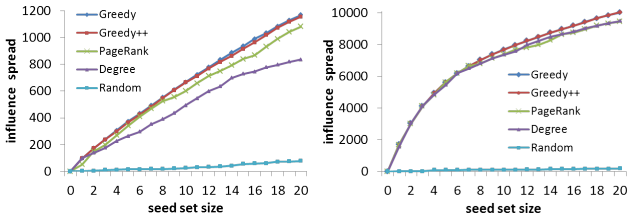
In the linear case, consider a random walk. It can start with any node v in the graph. In each step, it randomly move toward an

[2] http://research.microsoft.com/en-us/people/weic/graphdata.zip
[3] http://snap.stanford.edu/data



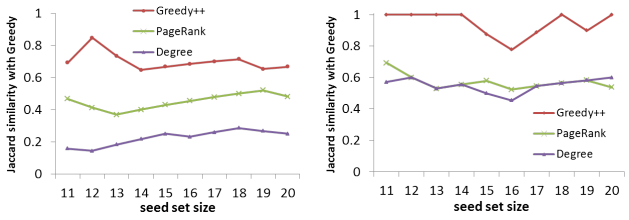
(a) Linear Threshold (b) Concave Threshold (c) Convex Threshold (d) Majority Vote

Figure 2: Influence spread of various algorithms in NetHEPT, with different distribution of δ_u. ($X \sim U[0,1]$.) (a) $\delta_u = X$ (submodular). (b) $\delta_u = X^2$ (submodular). (c) $\delta_u = \sqrt{X}$ (nonsubmodular). (d) $\delta_u = 0.5$ (nonsubmodular).



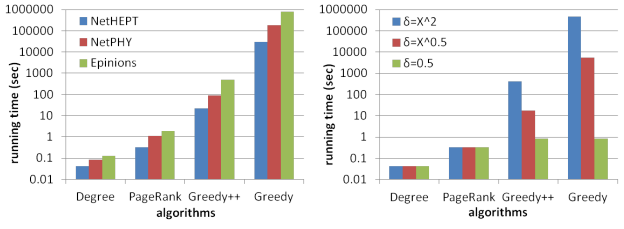
(a) NetPHY (b) Epinions

Figure 3: Influence spread of various algorithms in (a) NetPHY and (b) Epinions. ($F_\delta(x) = x$.)



(a) NetPHY (b) Epinions

Figure 4: Jaccard similarity of the seed set with Greedy in (a) NetPHY and (b) Epinions. ($F_\delta(x) = x$.)



(a) (b)

Figure 5: (a) Running time of various algorithms on three datasets. ($F_\delta(x) = x$.) (b) Running time of various algorithms in NetHEPT, with different distribution of δ_u. ($X \sim U[0,1]$.)

"in-degree" neighbor (which means the movement is reverse to the direction of the edge). The random walk will not stop until one node has been visited twice. Note that this kind of random walk is very similar to the one in PageRank except the stop condition. Borgs et al. [1] define the set of the nodes which are visited during the random walk as the *RR (Reverse Reachable) Set* for v, and they prove that in the linear threshold case, the set cover problem of RR sets is equivalent to the influence maximization problem. Therefore the probability for a node to be included in the RR sets reflects its influence power. This fact explains why PageRank (or the "inverse random walk") is useful.

But in nonlinear cases like the Figure 2(b), since most nodes in the network have a low threshold, the first person we select can affect a wide range of nodes. But when we want to choose other influential nodes, PageRank and Degree tend to select "central" nodes with high degree. But most of the "central" nodes can easily be affected from the first node. (Because in an undirected graph, nodes with larger "out-degree" also have larger "in-degree", or more chances to be affected.) So they are no longer useful in the spreading process after the most influential node being selected.

We also compute the Jaccard similarity between the seed set selected by Greedy and that by other algorithms. The result is shown in Figures 4(a) and 4(b), from which we can see that Greedy++ consistently shares much higher similarity with Greedy than PageRank and Degree do, meaning the introduction of LazyForward and StaticGreedy strategies will not cause significant loss of effectiveness.

Efficiency. We now test the running time of these algorithms. Figure 3 shows our experimental results.

As we expected, Greedy++ runs consistently faster than Greedy, with more than three orders of magnitude speedup. For example, in the linear threshold case, it takes Greedy more than 9 days to get the top-20 influential nodes in Epinions while Greedy++ only requires 8 minutes.

In the concave threshold case, Greedy++ spends more time because δ_u is small and the influence spread tends to be wide. But it is worthwhile because the strategies only finding "central nodes" no longer work in this case (see Figure 2(b)). In Majority Vote model, the efficiency of the greedy algorithm dramatically rises because the estimation of influence spread becomes easy.

5 CONCLUSIONS

In this paper, we have discussed how to find top-K influential nodes in social networks under a game theoretic model. We show the hardness of the optimization problem itself, as well as the hardness of

calculating the objective function. We prove the approximation guarantee of the greedy algorithm under necessary assumptions. We also accelerate our algorithm with the combination of Lazy-Forward and StaticGreedy. Our experimental results demonstrate that `Greedy++` matches `Greedy` in the spreading effect while significantly reducing running time, and it outperforms other heuristic algorithms such as MaxDegree and PageRank.

REFERENCES

[1] C. Borgs, M. Brautbar, J. Chayes, and B. Lucier. Maximizing social influence in nearly optimal time. In *SODA'14*, pages 946–957. SIAM, 2014.
[2] N. Chen. On the approximability of influence in social networks. In *SODA'09*, pages 1029–1037. SIAM, 2009.
[3] W. Chen, C. Wang, and Y. Wang. Scalable influence maximization for prevalent viral marketing in large-scale social networks. In *KDD'10*, pages 1029–1038. ACM, 2010.
[4] W. Chen, Y. Yuan, and L. Zhang. Scalable influence maximization in social networks under the linear threshold model. In *ICDM'10*, pages 88–97. IEEE, 2010.
[5] S. Cheng, H. Shen, J. Huang, W. Chen, and X. Cheng. Imrank: Influence maximization via finding self-consistent ranking. In *SIGIR'14*, pages 475–484. ACM, 2014.
[6] S. Cheng, H. Shen, J. Huang, G. Zhang, and X. Cheng. Staticgreedy: solving the scalability-accuracy dilemma in influence maximization. In *CIKM'13*, pages 509–518. ACM, 2013.
[7] D. Easley and J. Kleinberg. *Networks, crowds, and markets: Reasoning about a highly connected world.* Cambridge University Press, 2010.
[8] M. T. Irfan and L. E. Ortiz. A game-theoretic approach to influence in networks. In *AAAI'11*, pages 688–694. AAAI, 2011.
[9] D. Kempe, J. Kleinberg, and É. Tardos. Maximizing the spread of influence through a social network. In *KDD'03*, pages 137–146. ACM, 2003.
[10] J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. VanBriesen, and N. Glance. Cost-effective outbreak detection in networks. In *KDD'07*, pages 420–429. ACM, 2007.
[11] S. Morris. Contagion. *The Review of Economic Studies*, 67:57–78, 2000.
[12] E. Mossel and S. Roch. Submodularity of influence in social networks: From local to global. *SIAM Journal on Computing*, 39(6):2176–2188, 2010.
[13] Y. Tang, Y. Shi, and X. Xiao. Influence maximization in near-linear time: a martingale approach. In *SIGMOD'15*, pages 1539–1554. ACM, 2015.
[14] L. G. Valiant. The complexity of enumeration and reliability problems. *SIAM Journal on Computing*, 8:410–421, 1979.

A EQUILIBRIUM OF THE GAME

As a digression, we discuss our model from the perspective of game theory. An important problem is to find the *pure-strategy Nash equilibrium* (PSNE) of the game [8]. In a PSNE, the strategy each node adopts is the best strategy toward its neighbors. Obviously, when all nodes adopt strategy A (or strategy B), the whole network will achieve a PSNE. But a more meaningful problem is that given the initial state of each node, whether the network will converge to a PSNE or not. We take an early step in this problem, studying the case of Majority Vote model.

As we mentioned, the spread of a new behavior will finally stop in at most $|V|$ steps under CG model. But we have to point out that the final state is not a PSNE because nodes in the initial seed set may not have chosen the best strategy. To discuss the PSNE, we need to allow initial seed nodes to change their choices again. Therefore, the model can be explained as follows: Initially some nodes in the network choose strategy A, while others choose B. In each step, each node decides whether to change its strategy according to the payoff-maximization principle. Under this model, nodes can switch their decision for many times. And once all nodes stop changing their states, our game achieves a PSNE. To simplify the model, we assume that no one will meet a dilemma in the game (i.e., $p_B x_B \neq p_A x_A$ at any time[4]).

[4]e.g., $p_A = p_B = 1$ and each node has an odd number of neighbors.

But will this equilibrium always appear? Not really. Let us consider a complete graph with 4 nodes. Initially two nodes choose A and the other two choose B, then all of the four nodes will "swing" between A and B forever. And the repeated game will become a "2-periodic" process, thus will never converge to a PSNE. Actually we have the following conclusion:

Lemma A.1. *$\forall p_A, p_B \in \mathbb{Z}^+$, the repeated game will either go to a PSNE or become a "2-periodic" process in $O(\max\{p_A, p_B\}|E|)$ steps.*

Proof. We can assume that $p_A \geq p_B$. In step k, let $f_k(v) = p_A$ if node v adopts strategy A, and let $f_k(v) = -p_B$ if v adopts B. We define the potential function as

$$\begin{aligned} F_k &= \sum_{(u,v)\in E} (f_k(u)f_{k-1}(v) + f_{k-1}(u)f_k(v)) \\ &= \sum_u \sum_{v\in N(u)} f_k(u)f_{k-1}(v) = \sum_u \sum_{v\in N(u)} f_k(v)f_{k-1}(u). \end{aligned} \tag{16}$$

Therefore

$$\begin{aligned} F_{k+1} - F_k &= \sum_u \sum_{v\in N(u)} f_{k+1}(u)f_k(v) - \sum_u \sum_{v\in N(u)} f_k(v)f_{k-1}(u) \\ &= \sum_u (f_{k+1}(u) - f_{k-1}(u))(\sum_{v\in N(u)} f_k(v)). \end{aligned} \tag{17}$$

Since $p_B x_B \neq p_A x_A$ at any time, we have $\sum_{v\in N(u)} f_k(v) \neq 0$. If $\sum_{v\in N(u)} f_k(v) > 0$, u should choose A in the next step and therefore $f_{k+1}(u) = p_A \geq f_{k-1}(u)$. If $\sum_{v\in N(u)} f_k(v) < 0$, u should choose B in the next step and therefore $f_{k+1}(u) = -p_B \leq f_{k-1}(u)$. In summary, $f_{k+1}(u) - f_{k-1}(u)$ always has the same sign with $\sum_{v\in N(u)} f_k(v)$. Therefore $F_{k+1} - F_k \geq 0$.

It is also easy to prove that:

(1) $F_k \leq 2p_A^2|E|$, $F_k \geq -2p_A p_B|E|$.
(2) If $F_{k+1} - F_k \neq 0$, then $F_{k+1} - F_k \geq (p_A + p_B) \times 1$.

So $\exists K \leq \frac{2p_A^2|E| - (-2p_A p_B|E|)}{p_A + p_B} = O(p_A|E|)$ such that $F_{K+1} - F_K = 0$, or

$$\sum_u (f_{K+1}(u) - f_{K-1}(u))(\sum_{v\in N(u)} f_K(v)) = 0. \tag{18}$$

Since $\sum_{v\in N(u)} f_K(v) \neq 0$, we have $f_{K+1}(u) - f_{K-1}(u) = 0$ ($\forall u \in V$). So each node makes the same choice in step $K-1$ and step $K+1$. Therefore, they will make the same choice in step K and step $K+2$, and so on. The process then has a period of 1 or 2, corresponding to a PSNE or a "2-periodic" process respectively. □

With the help of Lemma A.1, we can have an efficient algorithm computing PSNE. We just simulate the evolution of each node's state. Once we find that the process becomes "2-periodic" (it only takes 2 more steps), we know that the network cannot achieve a PSNE. Otherwise we can get the PSNE in $O(\max\{p_A, p_B\}|E|)$ steps. The time complexity of the algorithm is $O(\max\{p_A, p_B\}|E|(|V| + |E|))$. We conclude the result in Theorem A.2.

Theorem A.2. *Suppose $p_A, p_B \in \mathbb{Z}^+$ and are fixed, given the initial state of each node, it is **polynomial-time** to answer the following questions: (1) Will the network converge to a PSNE? (2) If so, compute the PSNE.*