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 individual makes its decision based on the benefit of coordination with its 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 the motivation to use a small subset of influential individuals in a social network to activate a large number of people. Kempe et al. [7] 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 [6] 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 [7]. 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 a 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 [1] and Linear Threshold model [7]. 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* [10], 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 $(1 - 1/e - \epsilon)$-approximation solution.

As a traditional method, Kempe et al. [7] 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 [8] and StaticGreedy [5]. 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. [7] 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` [8]. Other researchers focus on how to calculate the influence spread efficiently. For instance, Chen

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

ACM ISBN 978-1-4503-5022-8/17/08…$15.00.
https://doi.org/10.1145/3077136.3080709

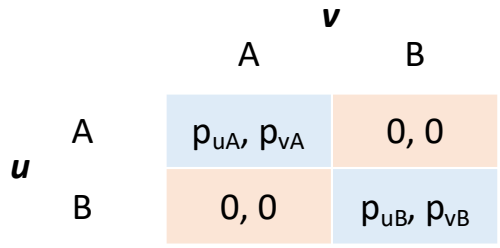


Figure 1: Payoff matrix of the coordination game.

et al. [2, 3] use arborescences or DAGs to represent the original graph. Cheng et al. propose a `StaticGreedy` strategy [5] and a self-consistent ranking method [4].

Morris [9] is the first to propose a coordination game model in contagion. This model is also discussed detailedly in Easley and Kleinberg's textbook [6]. 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 adopts A than that when it adopts B, it will choose strategy A. Otherwise, it will choose strategy B.

According to the actual situation, we have the following assumptions about 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

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

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 [1].

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

$$\begin{gathered} f_B(x) = \frac{\alpha}{(x+1)^\gamma} \ (x \geq 0); \\ \text{where } \gamma > 1 \text{ and } \alpha = \frac{1}{\int_0^\infty \frac{1}{(x+1)^\gamma}\mathrm{d}x} = \gamma - 1; \end{gathered} \tag{5}$$

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.*

The hardness result directly follows the NP-hardness of Influence Maximization under Majority Vote model [1] and LT model [7] and the #P-hardness of computing the objective function under LT model [3].

3 ALGORITHMS

Submodularity. To find a greedy algorithm with an approximation guarantee, the submodularity of the objective function is necessary.

We first recall the general diffusion process defined by Mossel and Roch in [10].

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 $t \geq 1$,

$$S_t = S_{t-1} \cup \{v | v \in V - S_{t-1} \ \wedge \ f_v(S_{t-1}) \geq \theta_v\}: \tag{7}$$

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

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

$$\theta_v = F_\delta(\delta_v) \text{ and } f_v(S) = F_\delta\left(\frac{|S \cap N(v)|}{\deg(v)}\right): \tag{8}$$

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

$$\Pr[F_\delta(\delta_v) \leq x] = \Pr[\delta_v \leq F_\delta^{-1}(x)] = F_\delta(F_\delta^{-1}(x)) = x: \tag{9}$$

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

$$\begin{aligned} f_v(S) \geq \theta_v &\iff F_\delta\left(\frac{|S \cap N(v)|}{\deg(v)}\right) \geq \theta_v \\ &\iff |S \cap N(v)| \geq F_\delta^{-1}(\theta_v)\deg(v) \\ &\iff |S \cap N(v)| \geq \delta_v \deg(v): \end{aligned} \tag{10}$$

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 $[0; 1]$.*

It is not difficult to understand Lemma 3.1 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 positive skewness, or people tend to have a higher evaluation of 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 [10] have proved that $\sigma(S_0)$ is monotone and submodular if and only if f_v is monotone and submodular for any $v \in V$. Therefore, we can get Theorem 3.2 immediately.

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

Theorem 3.2 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 $F_\delta(x) = \mathsf{I}(x \geq \delta)$ is not concave in $[0; 1]$, where $\mathsf{I}(\cdot)$ 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 [7], the greedy algorithm given in Algorithm 1 (`Greedy`) returns a $(1 - 1/e - \epsilon)$-approximate solution. 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 strategies - LazyForward [8] and StaticGreedy [5] to accelerate our algorithm. The reasons why they are useful in submodular cases have been explained in [8] and [5] respectively.

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.

Instead of conducting a huge number of Monte Carlo simulations each time, we generate a rather small number of snapshots at the very beginning. In all the iterations, we run simulations on these snapshots and use the average to estimate the objective function.

We name the accelerated algorithm as `Greedy++`.

Algorithm 1 `Greedy`(k, σ)

```
initialize S_0 = ∅
for i = 1 to k do
    select u = arg max_{v ∈ V − S_0} (σ(S_0 ∪ {v}) − σ(S_0))
    S_0 = S_0 ∪ {u}
end for
output S_0
```

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`[1], and one online social network `Epinions`[2]. 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

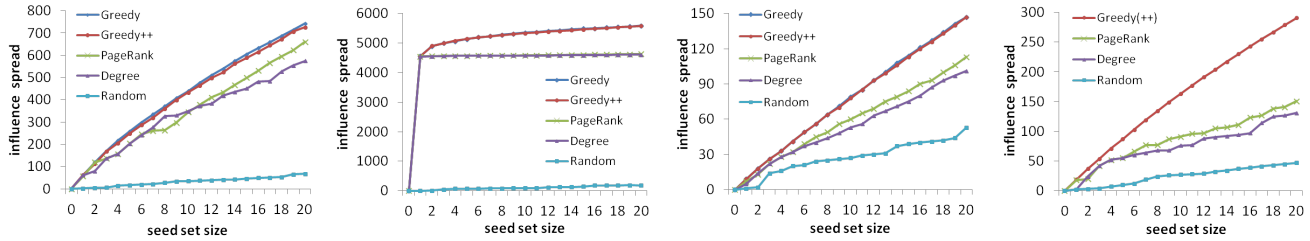
Algorithms. A total of five algorithms are tested. Besides `Greedy` and `Greedy++` proposed in this paper, we use other three heuristic algorithms as benchmark methods.

(1) `PageRank` chooses nodes with the largest PageRank value. For directed networks, influential nodes are considered to have a large number of out-links, while nodes with high PageRank values are considered to have lots of in-links. Therefore, in Epinions, we first change the direction of all edges in the graph and then run PageRank. We use $\alpha = 0.9$ as the random jump parameter.

(2) `Degree` chooses nodes with the largest out-degree.

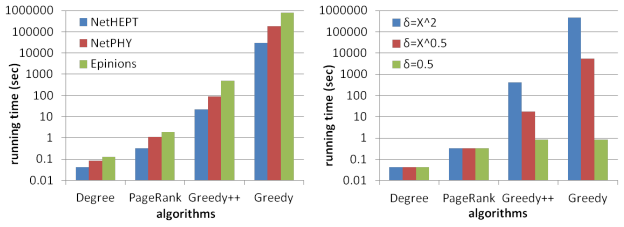
(3) `Random` chooses nodes at random.

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



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

Figure 2: Influence spread of various algorithms on 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) (b)

Figure 3: (a) Running time of various algorithms on three datasets with $F_\delta(x) = x$. (b) Running time of various algorithms on NetHEPT with different distributions of δ_u ($X \sim U[0;1]$:)

There are several other efficient algorithms to solve influence maximization under IC model or LT model, such as PMIA [2], LDAG [3] and IMM [11]. However, 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, the distribution of δ_u can be various. We run influence maximization algorithms under four different spreading models where δ_u is X, X^2, $\sqrt{X}$ and 0:5, respectively ($X \sim U[0;1]$). Accordingly, the distribution function $F_\delta(x)$ is x, $\sqrt{x}$, x^2 and $\mathbb{I}(x \geq 0{:}5)$.

Figure 2 shows our experimental results on NetHEPT. In Figure 2, Greedy++ consistently performs on par with Greedy and significantly outperforms other heuristic algorithms in all cases. According to Theorem 3.2, the first two cases are submodular, while the other two are not. However, our experimental results indicate that Greedy and Greedy++ still perform well in the non-submodular cases. In two larger graphs NetPHY and Epinions, we get similar experimental results.

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 on Epinions, while Greedy++ only requires about 8 minutes.

In the concave threshold case, Greedy++ spends more time because δ_u is small and the influence spread tends to be wide. However, this is worthwhile because the strategies only finding "central nodes" no longer work in this case (see Figure 2(b)).

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 reduces running time, and it outperforms other heuristic algorithms such as MaxDegree and PageRank.

Acknowledgements. This work is supported by 973 Program under Grant No.2014CB340405, NSFC under Grant No.61532001 and No.61370054. We thank the anonymous reviewers for their valuable comments.

REFERENCES

[1] N. Chen. 2009. On the Approximability of Influence in Social Networks. In *SODA'09*. SIAM, Austin, Texas, USA, 1029–1037.
[2] W. Chen, C. Wang, and Y. Wang. 2010. Scalable influence maximization for prevalent viral marketing in large-scale social networks. In *KDD'10*. ACM, Washington, DC, USA, 1029–1038.
[3] W. Chen, Y. Yuan, and L. Zhang. 2010. Scalable influence maximization in social networks under the linear threshold model. In *ICDM'10*. IEEE, Sydney, Australia, 88–97.
[4] S. Cheng, H. Shen, J. Huang, W. Chen, and X. Cheng. 2014. Imrank: Influence maximization via finding self-consistent ranking. In *SIGIR'14*. ACM, Gold Coast, Australia, 475–484.
[5] S. Cheng, H. Shen, J. Huang, G. Zhang, and X. Cheng. 2013. Staticgreedy: solving the scalability-accuracy dilemma in influence maximization. In *CIKM'13*. ACM, San Francisco, CA, USA, 509–518.
[6] D. Easley and J. Kleinberg. 2010. *Networks, crowds, and markets: Reasoning about a highly connected world*. Cambridge University Press.
[7] D. Kempe, J. Kleinberg, and É. Tardos. 2003. Maximizing the spread of influence through a social network. In *KDD'03*. ACM, Washington, DC, USA, 137–146.
[8] J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. VanBriesen, and N. Glance. 2007. Cost-effective outbreak detection in networks. In *KDD'07*. ACM, San Jose, CA, USA, 420–429.
[9] S. Morris. 2000. Contagion. *The Review of Economic Studies* 67 (2000), 57–78.
[10] E. Mossel and S. Roch. 2010. Submodularity of influence in social networks: From local to global. *SIAM J. Comput.* 39, 6 (2010), 2176–2188.
[11] Y. Tang, Y. Shi, and X. Xiao. 2015. Influence maximization in near-linear time: a martingale approach. In *SIGMOD'15*. ACM, Melbourne, Australia, 1539–1554.
[12] L. G. Valiant. 1979. The complexity of enumeration and reliability problems. *SIAM J. Comput.* 8 (1979), 410–421.

A PROOF OF THEOREM 2.1

Proof. (1) Chen [1] 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. [3] 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.[3] 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 [12]. Let $n = |V|$ and $D = \max_{v \in V} \deg_{in}(v)$. From G, we construct $n + 1$ graphs $G_1; G_2; \cdots; G_{n+1}$. To get G_i $(1 \le i \le n + 1)$, 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 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 [3], we have

$$\sigma_{G_i}(S_0) = \sum_{\pi \in \mathcal{P}} \prod_{e \in \pi} w_i; \quad (1 \le i \le n + 1); \tag{11}$$

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

$$\sigma_{G_i}(S_0) = \sum_{j=0} \sum_{\pi \in B_j} \prod_{e \in \pi} w_i = \sum_{j=0} \sum_{\pi \in B_j} w_i^j = \sum_{j=0} w_i^j |B_j|: \tag{12}$$

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

Finally, we notice that for each $j = 1; 2; \cdots; n$, 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. □

B PROOF OF LEMMA 3.1

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 $a; b; \lambda$ respectively, we will have

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

Let $X_i = (\frac{i}{Mq}; F_\delta(\frac{i}{Mq}))$ be the points on the curve of F_δ $(i = N_1 q; \cdots; N_2 q)$ 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_2+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

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

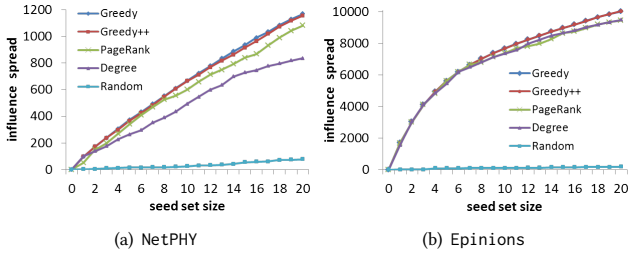
which violates the submodularity of f_v. □

C THE GREEDY++ ALGORITHM

Algorithm 2 Greedy++(k, σ, R')

```
initialize S_0 = ∅
for i = 1 to R' do
    generate the threshold δ_v (∀v ∈ V) for snapshot G_i
end for
for all v ∈ V do
    Δ_v = +∞ //initialize the marginal gain of each node
end for
for i = 1 to k do
    for all v ∈ V − S_0 do
        cur_v = False
    end for
    while True do
        u = arg max_{v∈V−S_0} Δ_v //maintain a priority queue
        if cur_u then
            S_0 = S_0 ∪ {u}
            break
        else
            Δ_u = (1/R') Σ_{i=1}^{R'} (σ_{G_i}(S_0 ∪ {u}) − σ_{G_i}(S_0))
            reinsert u into the priority queue and heapify
            cur_u = True
        end if
    end while
end for
output S_0
```

D ADDITIONAL EXPERIMENTAL RESULTS



(a) NetPHY (b) Epinions

Figure 4: Influence spread of various algorithms on (a) NetPHY and (b) Epinions. ($F_\delta(x) = x$.)

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