A Game Theoretic Model for Influence Maximization

Yu Zhang
Dept. of Computer Science
Peking University
Beijing, China
yutazh@pku.edu.cn

Yan Zhang
Dept. of Machine Intelligence
Peking University
Beijing, China
zhy@cis.pku.edu.cn

ABSTRACT

Influence maximization, aiming to find top-K influential nodes under certain spreading models, is considered to be the algorithmic basis of viral marketing. However, most researches discuss this problem under Independent Cascade (IC) model or Linear Threshold (LT) model. In this paper, we study influence maximization from the perspective of game theory. We propose a Coordination Game (CG) model, in which every individuals make their decisions based on the benefit of coordination with their network neighbors, to study information propagation. We first discuss two special cases of our model: Majority Vote and Linear Threshold and prove some hardness results. We also study how to compute Nash equilibrium given the initial state of each node. Then we prove the monotonicity and submodularity of the objective function when the threshold distribution is concave. Thus the greedy algorithm can return a $(1-1/e-\epsilon)$-approximate solution. Finally we combine two strategies - LazyForward and StaticGreedy - to accelerate the algorithm. We compare our algorithms with other baseline heuristics. Experimental results show that after using the acceleration strategies, the accuracy of our algorithm is on par with the greedy algorithm's and better than others', while it is three orders of magnitude faster than the greedy algorithm.

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 describe relationships between individuals in our society, and 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. Influence maximization, aiming to find top-K influential nodes under certain spreading models, has been extensively researched. Kempe et al. [20] build a theoretical framework of influence maximization by transforming it into a submodular optimization problem. 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.

In most spreading models, each node has two states: active and inactive. Equivalently saying, it has two choices. We can model information diffusion as the process of individual decision-making. As individuals make their decisions based on choices of their neighbors, a particular pattern of behavior can begin to spread across the links of the network. Easley and Kleinberg [14] 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. Our model is based on a game. As we all know, game theory provides an accurate description of people's behavior in decision-making. Morris [25] uses a coordination game on every edge in the social network to model people's intention to make the same choice with most of their network neighbors. This is the prototype of our Coordination Game (CG) model[1].

Influence maximization under CG model is useful in viral marketing. Let us recall the example in [20]. 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 and give up the old 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 brings a lot of inconvenience and reduce their benefits. (e.g. People using different operating systems in their computers may have some problems with compatibility when sharing files. Users from different kinds of social network software cannot communicate with each other timely, etc.) Our CG model can describe these two reasons precisely.

In this paper, we study how to find Top-K influential nodes under CG model. Firstly, we propose our CG model and discuss 2 special cases of it: Majority Vote model [3] and the well-known Linear Threshold model [20]. Under Major Vote model, we prove the NP-hardness of influence maximization and propose a polynomial-time algorithm to find

© 2017 ACM. ISBN 978-1-4503-2138-9.
DOI: 10.1145/1235

[1]We will explain the details of the model in Section 3.

the *pure strategy Nash equilibrium* (PSNE). Under Linear Threshold model, we prove the #P-hardness of computing the objective function, which has slight difference from Chen et al.'s result [8]. By discussing the hardness of special cases, we naturally get the hardness of the general case of our CG model. Then we try to find a good approximation algorithm for the problem. We embed our CG model into the scenario of general diffusion process defined by Mossel and Roch in [26]. And with the help of the main result in [26], we prove the monotonicity and submodularity of the objective function if and only if the *cumulative distribution function* (CDF) of people's threshold is concave (or intuitively, people tend to have a higher evaluation on new products than old ones). Therefore the greedy algorithm can return a $(1-1/e-\epsilon)$ approximate solution. For the nonsubmodular case, recall that it is NP-hard to approximate the influence maximization problem within a factor of $n^{1-\epsilon}$ in general [20].

As a traditional method, Kempe et al. [20] use 10,000 times of Monte Carlo simulations to approximate the objective function every time, but it takes us too much time on large-scale networks. In our experiment, the greedy algorithm takes over 9 days to find the Top-20 influential nodes in a network with 75K nodes. To accelerate our algorithm, we use two efficient heuristics - LazyForward [23] and StaticGreedy [10] to reduce the times of simulations as much as possible. We compare our `CGGreedy` and `CGSL` algorithm with other heuristics, such as MaxDegree or PageRank in four real-world networks. Experimental results show that `CGSL` algorithm can activate more nodes than other heuristics in all the datasets and can achieve influence effect on par with the greedy algorithm. In fact, the seed sets chosen by `CGSL` and the greedy algorithm share high similarity. And `CGSL` runs faster than the greedy algorithm by three orders of magnitude. For example, `CGSL` only takes about 8 minutes to find the Top-20 influential nodes in the network with 75K nodes.

To summarize, we make the following contributions:

- From the perspective of game theory, we propose a coordination game model to study influence maximization and discuss how to compute PSNE in some special cases.
- From the perspective of complexity, we prove the NP-hardness of influence maximization problem and the #P-hardness of computing the objective function under CG model.
- From the perspective of algorithms, we prove the monotonicity and submodularity of the objective function under certain assumptions, and propose a greedy algorithm with an approximation ratio of $(1-1/e-\epsilon)$ under CG model.
- We use two strategies to accelerate our algorithm. Experimental results on real datasets show that our method is both effective and efficient, even for nonsubmodular cases.

The rest of this paper is organized as follows. In Section 2, we review related works. In Section 3, we propose our CG model and discuss two special cases: Majority Vote and Linear Threshold. In Section 4, we prove the monotonicity and submodularity of the objective function under certain assumptions and propose the accelerating strategies. Then in Section 5, we show our experimental results. Finally, we conclude the paper in Section 6.

2. RELATED WORK

Influence maximization is first studied by Domingos and Richardson [12, 31] from the probabilistic perspective. Kempe et al. [20] build a great algorithmic framework of it by transforming it into a discrete optimization problem. They give a greedy algorithm with $(1-1/e-\epsilon)$ approximation rate under IC model and LT model. This rate is guaranteed by the monotonicity and submodularity of the global objective function. They also conjecture that the submodularity of the "local" influence function indicates the submodularity of the "global" objective function. Mossel and Roch prove this conjecture in [26] based on a novel antisense coupling technique. There are some other works broadening the class of spreading models, such as General Threshold model [21] and Continuous-Time IC model [15]. And optimizing the objective function on multiple spreading models "simultaneously" is studied in [5, 17] recently.

Morris [25] is the first to propose the Coordination Game model in contagion. He shows that coordination is the cause of cascades and clusters are obstacles to cascades. Easley and Kleinberg use a whole chapter to give more extended discussions about CG model in their textbook [14] to explain cascading behaviors in networks. Irfan and Ortiz [18] introduce a game-theoretic approach to study influence in networks. They mainly focus on the PSNE of the game. Dubey et al. [13] focus on the competing game of different products to dominate the market.

Over the past few years, a lot of efforts have been made on efficient computing methods of the objective function because the basic greedy algorithm takes too many times of Monte Carlo simulation. These work can be divided into two directions. On one direction, researchers try to reduce the number of trials that need Monte Carlo simulations [23, 16, 36]. For example, Leskovec et al. [23] propose a "Lazy-Forward" strategy based on the submodularity of the spreading function. Their `CELF` algorithm effectively reduces the candidate nodes to simulate in each round. On the other direction, researchers focus on how to calculate the influence spread efficiently [7, 6, 8, 35, 10, 9, 24]. For example, Chen et al. [7, 6, 8] use various heuristics to simplify the graphs on which they conducted Monte Carlo simulations. They use arborescences or DAGs to represent the original graph. Cheng et al. [10] give a "Static-Greedy" strategy. Instead of conducting a huge number of Monte Carlo simulations each time, they produce a rather small number of snapshots at the very beginning, and use the same set of snapshots in all iterations. Some of the acceleration strategies mentioned above can be adopted in our CG model directly, and we will explain the details in Section 4.

Recently, some efficient randomized algorithms are proposed to solve the problem. Borgs et al. [2] make a theoretical breakthrough and present an $O(kl^2(m+n)\log^2 n/\epsilon^3)$-time algorithm under IC model. Their *Reverse Influence Sampling* method returns a $(1-1/e-\epsilon)$-approximate solution with at least $(1-n^{-l})$ probability. Tang et al. [33, 32] improve this result with $O((k+l)(m+n)\log n/\epsilon^2)$ time complexity using a martingale approach, making it efficient in billion-scale networks. Nguyen et al. [29] further accelerate the algorithm through a *Stop-and-Stare* strategy.

3. PROBLEM FORMULATION

3.1 Coordination Game Model

In a social network $G=(V,E)$, we study a situation in

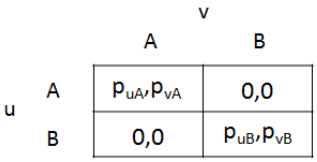


Figure 1: Payoff matrix of the coordination game.

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 $(u, v) \in E$,[2] in which players u and v both have two strategies A and B. The payoffs are defined as follows:

- if u and v both adopt strategy A, they will get payoffs $p_{uA} > 0$ and $p_{vA} > 0$ respectively;
- if they both adopt strategy B, they will get payoffs $p_{uB} > 0$ and $p_{vB} > 0$ respectively;
- if they adopt different strategies, they each get a payoff of 0.

We can write this in terms of a payoff matrix, as 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:

- All p_{uA} and p_{uB} $(u \in V)$ may not be equal to each other. The payoffs depend both on users and products. Since each person in the social network values behaviors A and B differently, their payoff functions in the coordination game will be different as well.
- p_{uA} and p_{uB} $(u \in V)$ can either be constants or random variables. As we all know, the cascading behaviors in networks are always considered to have determinate principles with some stochastic factors. In our model, all p_{uA} $(u \in V)$ are independent and identically distributed random variables. So are all p_{uB} $(u \in V)$.

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. What will u do to maximize its payoff? Obviously, it will adopt B if and only if

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

or equivalently,

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

Influence Maximization Problem. Suppose now the market is dominated by A (i.e., all 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. It is not hard to proof that node who switches to A at some point during this process will never switch back to B at a later point. So we can regard the evolution of nodes' choices as a spreading process of B in the network. And the spread of behavior B will finally stop in at most $n = |V|$ steps. Because when there are no new nodes switching to B at some point, there will not be any forever.

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 the endpoint, or

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

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

The model proposed by us seems to be quite simple. But we will show its adaptability by discussing two special cases of our model, which are famous spreading models.

3.2 Special Case I: Majority Vote Model

Let us consider the simplest case, in which all p_{uA} $(u \in V)$ are constants and are equal to each other[3]. So are all p_{uB} $(u \in V)$. Equivalently, let

$$p_A = p_{uA}, \quad p_B = p_{uB}, \quad \forall u \in V.$$

Then each node in the network will adopt B if and only if

$$x_B \geq \frac{p_A}{p_A + p_B} deg(u) = \delta deg(u), \quad \delta \in [0, 1].$$

δ is a constant threshold same to all nodes. When $p_A = p_B$, or $\delta = \frac{1}{2}$, the spreading model is called Majority Vote model, which is extensively studied in [30]. Under this model, each node will have the same behavior with majority of its neighbors.

Hardness. Chen [3] prove the hardness of the influence maximization problem when $\delta = \frac{1}{2}$. We extend the result (by using a different reduction method) in Theorem 1.

THEOREM 1. *Influence maximization problem under Majority Vote model is NP-hard for any $0 < \delta < 1$, even for bipartite graphs.*

Equilibrium of the Game. Now 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 [18]. 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 this is not meaningful. We want to know that given the initial state of each node, whether the network will converge to a PSNE or not.

As is said before, 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 choose 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

[2]Unless otherwise specified, the graph discussed by us is undirected.

[3]Constants can be considered as a special case of random variables if we let them follow the "one-point" distribution. And from this perspective, being identically distributed means being equal to each other.

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 P-SNE. To simplify to 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]).

But will this equilibrium always appear? Not really. Let us consider a complete graph with 4 nodes (i.e., K_4). Initially two nodes choose A and the other two choose B, then all of the four nodes will "thrash" 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 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

$$F_k = \sum_{(u,v)\in E} (f_k(u)f_{k-1}(v) + f_{k-1}(u)f_k(v))$$
$$= \sum_u \sum_{v\in N(u)} f_k(u)f_{k-1}(v) = \sum_u \sum_{v\in N(u)} f_k(v)f_{k-1}(u).$$

Therefore

$$F_{k+1} - F_k = \sum_u \sum_{v\in N(u)} f_{k+1}(u)f_k(v) - \sum_u \sum_{v\in N(u)} f_k(v)f_{k-1}(u)$$
$$= \sum_u (f_{k+1}(u) - f_{k-1}(u))(\sum_{v\in N(u)} f_k(v)).$$

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 $F_{k+1} - F_k \geq 0$.

We can also 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

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

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 1, we can have an efficient algorithm for 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 2.

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

THEOREM 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.*

3.3 Special Case II: 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 (PDF) of p_{uB} is

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

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

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

If we set $\gamma = 2$, then $\Pr[\delta_u \leq x] = x (0 \leq x \leq 1)$. Therefore $\delta_u \sim U[0,1]$. And we get the second special case of our model, which is the well-known Linear Threshold model where the weights on the edges adjacent to node u is $1/deg(u)$ (i.e., $b_{vu} = \frac{1}{deg(u)}, \forall u, v \in V$).

Hardness. The NP-hardness of the influence maximization problem under LT model has been proved by Kempe et al. in [20]. And another interesting question is how to compute the exact value of $\sigma(S_0)$ when the spreading model is stochastic. Since the objective function is an expectation of some (continuous) random variables, it is impossible to enumerate all the cases. Chen et al. [8] 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.[5]

THEOREM 3. *It is #P-hard to compute $\sigma(S_0)$ under LT model, even in the special case where $b_{vu} = \frac{1}{deg(u)}$.*

4. GREEDY ALGORITHM

By discussing the hardness in some special cases, we naturally get the hardness in the general case of our CG model.

THEOREM 4. *(1) Influence maximization under CG model is NP-hard. (2) Computing the objective function under CG model is #P-hard.*

Now we begin to focus on the algorithmic perspective of CG model. As we all know, to find a greedy algorithm with approximation guarantee, the submodularity of the objective function is necessary.

4.1 Submodularity

We first recall the general diffusion process defined by Mossel and Roch in [26].

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 $S_0 \subseteq V$. After step $t \geq 1$,

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

[5]Note that $b_{vu} = const$ is not a special case of CG model. Therefore we have to show Theorem 3 as a way to prove the #P-hardness in general case.

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 also 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

$$\theta_v = F_\delta(\delta_v), \quad f_v(S) = F_\delta(\frac{|S \cap N(v)|}{deg(v)}).$$

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

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

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

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

LEMMA 2. *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 $g_v(S) = \frac{|S\cap N(v)|}{deg(v)}$. It is easy to prove that both F_δ and g_v are monotone increasing and submodular. So we can get that $f_v = F_\delta \circ g_v$ 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

$$\lambda F_\delta(a) + (1-\lambda)F_\delta(b) > F_\delta(\lambda a + (1-\lambda)b).$$

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

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

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

$$f_v(T \cup \{u\}) - f_v(T) = F_\delta(\frac{K_2+1}{Mq}) - F_\delta(\frac{K_2}{Mq}) = \frac{k(l_2)}{Mq}$$

$$> \frac{k(l_1)}{Mq} = F_\delta(\frac{K_1+1}{Mq}) - F_\delta(\frac{K_1}{Mq}) = f_v(S \cup \{u\}) - f_v(S),$$

which violates the submodularity of f_v. □

It is not difficult for us to understand Lemma 2 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 in comparison with the uniform distribution, which means most people in the network are easy to be influenced, or they always have a higher evaluation on new products than old ones. This assumption is reasonable in the real world, which can explain why we always tend to change our mobile phones. F_δ being continuous and strictly monotone increasing in $[0,1]$ is a technical assumption instead of an essential one. Most continuous distributions satisfy it. We define these two assumptions as the *concave threshold property*.

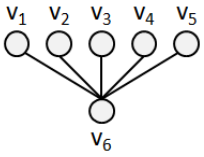


Figure 2: A counterexample of the submodularity of $\sigma(S_0)$ under Majority Vote model.

Note that if δ_u follows a continuous distribution in $[0,1]$, it will have a PDF $f_\delta = \frac{\mathrm{d}}{\mathrm{d}x}F_\delta$. And the concavity of F_δ is equivalent to the monotone decreasing property of f_δ.

For the general diffusion process, Mossel and Roch [26, 27] have proved that $\sigma(S_0)$ is monotone and submodular if and only if [6] f_v is monotone and submodular for any $v \in V$. Using this strong theorem together with Lemma 2, we can get Theorem 5 immediately.

THEOREM 5. *$\sigma(S_0)$ is monotone and submodular* ***iff*** *F_δ satisfies the concave threshold property.*

Theorem 5 is a strong tool to judge the objective function's submodularity under certain spreading models.

- 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 $\mathbb{I}()$ is the indicator function. The counterexample is also easy to find according to our proof. We have known that $x = \delta$ is a "non-concave" point of F_δ. Therefore we only need to find a case where the impact of the neighbors is "around" δ. For example, suppose $\delta = \frac{1}{2}$, we show the counterexample in Figure 2.

  If $S_0 = \{v_1\}$ or $\{v_1, v_2\}$ or $\{v_1, v_3\}$, we will have $S_n = S_0$. But if $S_0 = \{v_1, v_2, v_3\}$, we will have $S_n = \{v_1, v_2, v_3, v_4, v_5, v_6\}$. So $\sigma(\{v_1, v_2, v_3\}) - \sigma(\{v_1, v_2\}) = 6 - 2 > 2 - 1 = \sigma(\{v_1, v_3\}) - \sigma(\{v_1\})$.

- 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 assumptions. And these assumptions are necessary. Moreover, we have $\sigma(\emptyset) = 0$. Then by the famous result in [28], the greedy algorithm given in Algorithm 1 (`CGGreedy`) 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.

4.2 Speeding-Up Algorithm

As a traditional method, Kempe et al. [20] used $R = 10000$ times of Monte Carlo simulations to approximate the objective function every time. In one random simulation, the influence spread calculation from each node takes $O(|E|)$ time using Breadth-First Search. Therefore, the time complexity of `CGGreedy` is $O(kR|V||E|)$. It takes us too much

[6] The necessity is mentioned at the end of [27].

Algorithm 1 `CGGreedy`(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 `CGSL`(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

time on large-scale networks. Due to the hardness of computing $\sigma(S_0)$, we use two efficient heuristics - LazyForward [23] and StaticGreedy [10] to accelerate our algorithm as much as possible. There are lots of heuristics proposed in the past few years, but most of them depend on the "linear property" of IC model or LT model. Since F_δ can be various functions as long as it satisfies concave threshold property, CG model may not have such "linear property" in most cases. We find LazyForward and StaticGreedy applicable in CG model because they mainly use the submodularity instead of "linear property" to reduce time complexity.

Initially, we generate R' snapshots G_i ($i = 1, 2, ..., R'$) of G. For each seed set S_0, we run the simulation on these R' 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 $\sigma(S_0)$.

Aiming to find the node with the largest marginal gain of $\sigma(S_0)$ each time, 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.

We call this efficient heuristics `CGSL` (Static+Lazy). The time complexity of `CGSL` is $O(kR'|V||E|)$ if we ignore the factor of LazyForward. And with the LazyForward strategy, the time complexity of `CGSL` in practice will be smaller than we estimate. To store R' snapshots, we only need to store δ_u of each u in the network. So the space complexity of `CGSL` is $O(R'n)$. Our following experiment will show that `CGSL` with a small R' (e.g. $R' = 100$) can get similar results to `CGGreedy` with a large R (e.g. $R = 10000$). The reason is explained in [10].

5. EXPERIMENTS

To test the effectiveness and efficiency of our `CGGreedy` and `CGSL` algorithm, we conduct experiments on four real-world networks and compare our algorithms with other existing heuristics.

5.1 Experiment Setup

Datasets. The four real-world datasets include three collaboration networks and one online social network. Two of them, `GrQc` and `Epinions`, are downloaded from Jure Leskovec's Website [22]. The other two, `NetHEPT` and `NetPHY`, are downloaded from Wei Chen's Website [4]. In the three scientific collaboration networks, nodes represent authors and edges represent coauthor relationships among authors. `Epinions` is a who-trust-whom online social network in which a directed edge means that a user trusts another one. We summarize the statistical information of the four datasets in Table 1.

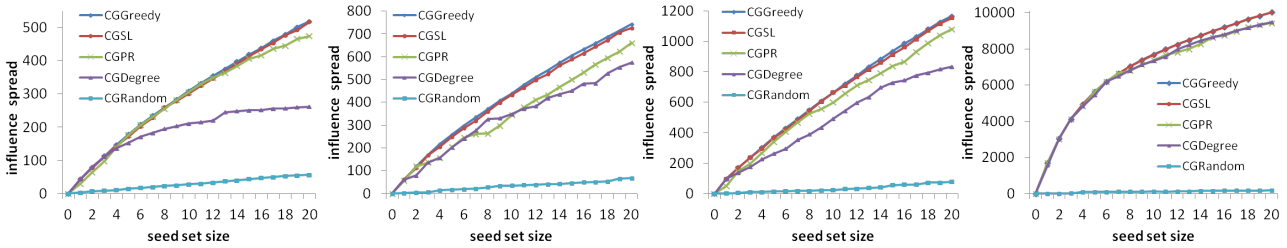
Datasets	$\|V\|$	$\|E\|$	Type
`GrQc`	5,242	14,496	Undirected
`NetHEPT`	15,233	58,991	Undirected
`NetPHY`	37,154	231,584	Undirected
`Epinions`	75,879	508,837	Directed

Table 1: Statistical information of four datasets

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

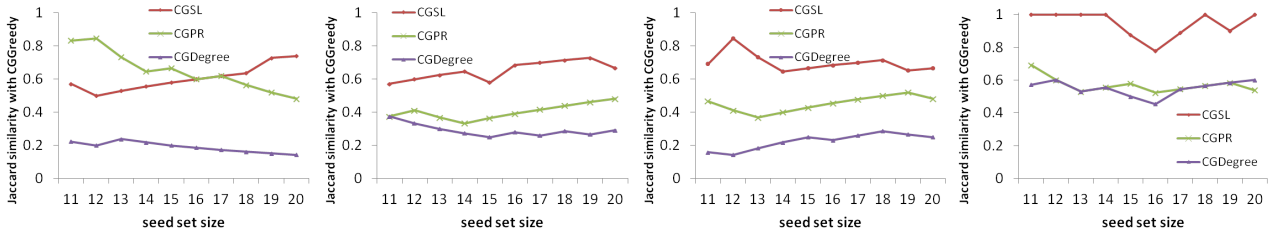
- `CGGreedy`. Algorithm 1 under CG model with the LazyForward optimization, using $R = 10000$ simulation runs for each influence estimation.
- `CGSL`. Algorithm 2 under CG model, using $R' = 100$ snapshots for influence estimation.
- `CGPR`. A baseline heuristic, choosing nodes with largest PageRank 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.
- `CGDegree`. A baseline heuristic, choosing nodes with largest out-degree.
- `CGRandom`. A baseline heuristic, choosing nodes at random.

There are a lot of other efficient algorithms to solve influence maximization problem under IC model or LT model, such as `PMIA` [6], `LDAG` [8], `TIM` [33], `IMM` [32] and `SSA` [29]. But they cannot be applied in CG model directly, and we will not put them into the comparison. All experiment code is written in C++, and all experiments are conducted on a server with 1.4GHz CPU and 64GB memory.



(a) GrQc (b) NetHEPT (c) NetPHY (d) Epinions

Figure 3: Influence spread of various algorithms in four datasets. ($F_\delta(x) = x$.)



(a) GrQc (b) NetHEPT (c) NetPHY (d) Epinions

Figure 4: Jaccard similarity of the seed set with CGGreedy on four datasets. ($F_\delta(x) = x$.)

5.2 Effectiveness Study

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

Linear Threshold Case. In the Linear Threshold case (i.e., $F_\delta(x) = x$), both CGGreedy and CGSL have the guarantee of accuracy. Figure 3 shows our experimental results of influence spread on four datasets by varying k from 1 to 20. From Figure 3, we can conclude that:

(1) CGSL consistently matches the performance of CGGreedy in all datasets. In fact, the two curves almost coincide in all four subfigures. And the difference of influence spread between CGSL and CGGreedy is 0.19%, 2.29%, 0.97% and 0.06% in four datasets respectively when $k = 20$.

(2) CGSL significantly outperforms other heuristic algorithms in the first three datasets. And in the Epinions dataset, the difference become smaller but still clearly visible in the figure. CGRandom performs badly in all cases.

(3) the curves of CGGreedy and CGSL are strictly concave in contrast to that of other heuristics, which also reflects the submodularity of our objective function.

Note that similarity in influence spread does not mean similarity in seed set selection. Since our goal is to find top-K influential nodes, not only influence spread but also seed set itself should be analyzed. So we compute the Jaccard similarity between the seed set selected by CGGreedy and that by other algorithms. We show the result in Figure 4, from which we can see that:

(1) An algorithm achieves high influential spread when it shares high similarity in seed set selection with CGGreedy. For example, in Epinions, both CGPR and CGDegree always have a Jaccard similarity higher than 0.45 with CGGreedy, so they do not show significant weakness in influence spread. In contrast, the similarity between CGDegree and CGGreedy never exceeds 0.4 in GrQc, NetHEPT and NetPHY, therefore CGDegree does not perform well in these three datasets. This result implies that CGGreedy may find some indeed influential nodes so that an algorithm can achieve high spreading effect as long as it makes the same choice with CGGreedy.

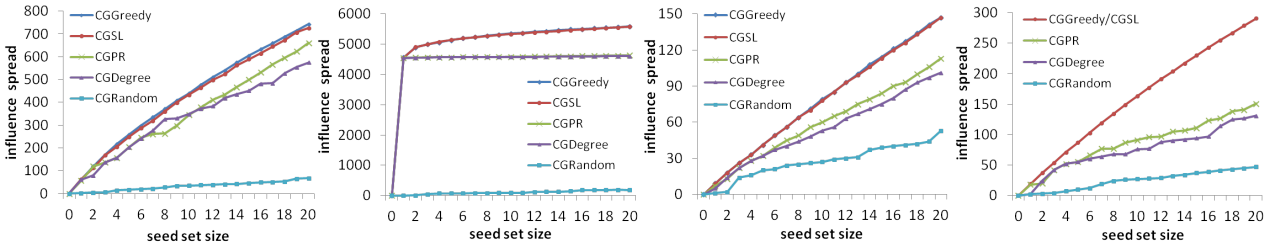
(2) CGSL consistently shares much higher similarity with CGGreedy than CGPR and CGDegree on all four networks. Note that in Epinions, the seed sets selected by these two algorithms are exactly the same when $k = 11 - 14$, 18 and 20. That means the introduction of LazyForward and StaticGreedy strategies will not cause significant loss of effectiveness. Besides, according to the conclusion mentioned above, CGSL may find more indeed influential nodes than CGPR and CGDegree.

Other Cases. In our CG model, distribution of δ_u can be various. Now we move from the linear case to some other cases. 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.

According to Theorem 5, it is obvious that the first two cases are submodular cases, while the other two are not. Figure 5 shows our experimental results. We need to point out that under Majority Vote model (which is a deterministic spreading model), objective function can be calculated efficiently. So the red curve in Figure 5(d) represents basic greedy algorithm without any optimization because LazyForward and StaticGreedy are not useful in that case.

From Figure 5, we find that CGSL still consistently matches the performance of CGGreedy and significantly outperforms other heuristic algorithms in all cases. Because of the non-submodularity of the spreading models, the approximation rate of CGGreedy and CGSL is not guaranteed in the third and the fourth cases. But our experimental results indicate that they still perform well in these cases. Besides, the curves of CGGreedy and CGSL remain concave in nonsubmodular cases.

In fact, if we consider the seed minimization problem [11] which is very similar to influence maximization, aiming to find the smallest seed set to activate the whole graph, we may get the reason why they perform well in non-submodular cases. Given a power-law network [1] with $\gamma > 2$, it is easy to prove that the greedy algorithm is an $O(1)$-approximation



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

Figure 5: 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).

algorithm under Majority Vote model with $\delta \geq \frac{1}{2}$.[7] Actually the conclusion is applicable to all the cases where $F_\delta(x) = 0$ ($x < 1/2$). Note that for the convex-threshold case, $F_\delta(x)$ is also small when $x < 1/2$.

CGPR and CGDegree perform even worse in nonlinear cases. For example, CGSL affects 20.60%, 30.09% and 97.56% more nodes than CGPR in the second, third and fourth cases. The huge difference in effectiveness of CGPR between linear and nonlinear cases 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 "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 CGPR except the stop condition. Borgs et al. [2] 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 CGPR (or the "inverse random walk") is useful.

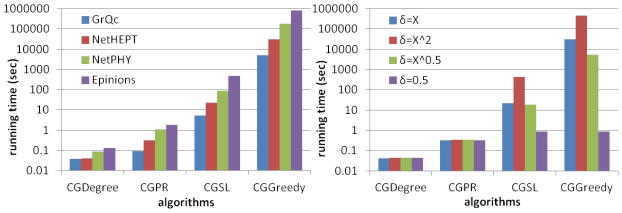
But in nonlinear cases like the second one, since most nodes in the network have a low threshold, the first person we select can affect a wide range of nodes. But when we want to choose other influential nodes, CGPR and CGDegree tend to select "central" nodes with high degree. But most of the "central" 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.

5.3 Efficiency Study

We now test the running time of CGGreedy, CGSL and other competing algorithms. Figure 6 shows our experimental results.

As we expected, CGSL runs consistently faster than CGGreedy, with more than three orders of magnitude speedup. For example, in the linear case, it takes CGGreedy more than 9 days to get the Top-20 influential nodes in Epinions while CGSL only requires 8 minutes. Even for the moderate sized dataset, i.e., NetHEPT and NetPHY, CGGreedy still requires 8.3 hours and 49.9 hours respectively. In contrast, CGSL only spends 22 seconds and 90 seconds on these two networks. More importantly, CGSL obtain the reduction of running time without affecting the guaranteed accuracy.

[7]The conclusion is derived from [11]. The proof is quite similar to that in [11] and we omit it.



(a) (b)

Figure 6: (a) Running time of various algorithms on four 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 other stochastic cases, our conclusion is still true. Note that CGSL will spend more time if δ_u is small, or the influence spread tends to be wide. And in Majority Vote model, the efficiency of the greedy algorithm dramatically rises because the estimation of influence spread becomes easy.

The reduction of running time CGSL achieves is far more than $R/R' = 100$ times. Our interpretation is that CGGreedy needs to generate $O(kR|V|^2)$ random numbers while CGSL only needs to generate $O(R|V|)$ ones, and in static snapshots, submodularity are better preserved so that we can make a bigger use of LazyForward strategy.

6. CONCLUSION AND FUTURE WORK

In this paper, we have discussed influence maximization problem 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 discuss how to compute PSNE of the coordination game and prove the monotonicity and submodularity of the objective function under certain assumptions. And these assumptions are necessary. Therefore the greedy algorithm has a guarantee of accuracy. We also propose a speed-up algorithm, CGSL, with the combination of LazyForward and StaticGreedy. Our experimental results demonstrate that CGSL matches the greedy algorithm in the spreading effect while significantly reducing running time. CGSL also outperforms other heuristic algorithms such as MaxDegree or PageRank.

There are some attractive challenges in the future. One direction is to find more effective and efficient heuristics and proving the result of inapproximability under CG model. Another direction is to explore more properties of the spreading process from the perspective of Nash Equilibrium, just like what we do in Section 3.2.

7. REFERENCES

[1] W. Aiello, F. Chung, and L. Lu. A random graph model for massive graphs. In *STOC'00*, pages 171–180. ACM, 2000.

[2] C. Borgs, M. Brautbar, J. Chayes, and B. Lucier. Maximizing social influence in nearly optimal time. In *SODA'14*, pages 946–957. SIAM, 2014.

[3] N. Chen. On the approximability of influence in social networks. In *SODA'09*, pages 1029–1037. ACM, 2009.

[4] W. Chen. http://research.microsoft.com/en-us/people/weic/graphdata.zip.

[5] W. Chen, T. Lin, Z. Tan, M. Zhao, and X. Zhou. Robust influence maximization. In *KDD'16*, pages 795–804. ACM, 2016.

[6] 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.

[7] W. Chen, Y. Wang, and S. Yang. Efficient influence maximization in social networks. In *KDD'09*, pages 199–208. ACM, 2009.

[8] 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.

[9] 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.

[10] 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.

[11] T. N. Dinh, D. T. Nguyen, and M. T. Thai. Cheap, easy, and massively effective viral marketing in social networks: truth or fiction? In *HT'12*, pages 165–174. ACM, 2012.

[12] P. Domingos and M. Richardson. Mining the network value of customers. In *KDD'01*, pages 57–66. ACM, 2001.

[13] P. Dubey, R. Garg, and B. De Meyer. Competing for customers in a social network: The quasi-linear case. In *WINE'06*, pages 162–173. Springer, 2006.

[14] D. Easley and J. Kleinberg. *Networks, crowds, and markets: Reasoning about a highly connected world*. Cambridge University Press, 2010.

[15] M. Gomez-Rodriguez, D. Balduzzi, and Bernhard Schölkopf. Uncovering the temporal dynamics of diffusion networks. In *ICML'11*, pages 561–568. ACM, 2011.

[16] A. Goyal, W. Lu, and L. Lakshmanan. Celf++: optimizing the greedy algorithm for influence maximization in social networks. In *WWW'11*, pages 47–48. ACM, 2011.

[17] X. He and D. Kempe. Robust influence maximization. In *KDD'16*, pages 885–894. ACM, 2016.

[18] M. T. Irfan and L. E. Ortiz. A game-theoretic approach to influence in networks. In *AAAI'11*, pages 688–694. AAAI, 2011.

[19] R. M. Karp. Reducibility among combinatorial problems. In *Proceedings of a symposium on the Complexity of Computer Computations*, pages 85–103. Plenum Press, New York, 1972.

[20] D. Kempe, J. Kleinberg, and É. Tardos. Maximizing the spread of influence through a social network. In *KDD'03*, pages 137–146. ACM, 2003.

[21] D. Kempe, J. Kleinberg, and É. Tardos. Influential nodes in a diffusion model for social networks. In *ICALP'05*, pages 1127–1138. Springer, 2005.

[22] J. Leskovec. http://snap.stanford.edu/data/.

[23] 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.

[24] Q. Liu, B. Xiang, E. Chen, H. Xiong, F. Tang, and J. X. Yu. Influence maximization over large-scale social networks: A bounded linear approach. In *CIKM'14*, pages 171–180. ACM, 2014.

[25] S. Morris. Contagion. *The Review of Economic Studies*, 67:57–78, 2000.

[26] E. Mossel and S. Roch. On the submodularity of influence in social networks. In *STOC'07*, pages 128–134. ACM, 2007.

[27] E. Mossel and S. Roch. Submodularity of influence in social networks: From local to global. *SIAM Journal on Computing*, 39(6):2176–2188, 2010.

[28] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher. An analysis of approximations for maximizing submodular set functions. *Mathematical Programming*, 14:265–294, 1978.

[29] H. Nguyen, M. Thai, and T. Dinh. Stop-and-stare: Optimal sampling algorithms for viral marketing in billion-scale networks. *SIGMOD'16*, pages 695–710, 2016.

[30] D. Peleg. Local majority voting, small coalitions and controlling monopolies in graphs: A review. In *SIROCCO'96*, pages 170–196, 1996.

[31] M. Richardson and P. Domingos. Mining knowledge-sharing sites for viral marketing. In *KDD'02*, pages 61–70. ACM, 2002.

[32] Y. Tang, Y. Shi, and X. Xiao. Influence maximization in near-linear time: a martingale approach. In *SIGMOD'15*, pages 1539–1554. ACM, 2015.

[33] Y. Tang, X. Xiao, and Y. Shi. Influence maximization: Near-optimal time complexity meets practical efficiency. In *SIGMOD'14*, pages 75–86. ACM, 2014.

[34] L. G. Valiant. The complexity of enumeration and reliability problems. *SIAM Journal on Computing*, 8:410–421, 1979.

[35] Y. Wang, G. Cong, G. Song, and K. Xie. Community-based greedy algorithm for mining top-k influential nodes in mobile social networks. In *KDD'10*, pages 1039–1048. ACM, 2010.

[36] C. Zhou, P. Zhang, J. Guo, X. Zhu, and L. Guo. Ublf: An upper bound based approach to discover influential nodes in social networks. In *ICDM'13*, pages 907–916. IEEE, 2013.

APPENDIX

A. PROOF OF THEOREM 1

PROOF. Let us consider the seed minimization problem under Majority Vote model: minimize $|S_0|$ subject to $|S_n| =$

$|V|$. This problem is the dual problem of influence maximization, aiming to find the smallest seed set to activate the whole graph. Our proof is divided into two parts. Firstly, we reduce the Vertex-Cover problem [19] to the seed minimization problem. Secondly, we reduce the seed minimization problem to our influence maximization problem.

In the first part, we consider three separate cases.

Case $\delta = \frac{1}{2}$. Given a graph $G = (V, E)$ and a constant k, we want to know whether G has a vertex cover of size at most k. We construct the following graph G'.

G' has $|V|+3|E|$ vertices. $|V|$ vertices $\{u_{11}, u_{12}, ..., u_{1|V|}\}$ in G' correspond to $|V|$ vertices $\{v_1, v_2, ..., v_{|V|}\}$ in G respectively. $|E|$ vertices $\{u_{21}, u_{22}, ..., u_{2|E|}\}$ in G' correspond to $|E|$ edges $\{e_1, e_2, ..., e_{|E|}\}$ in G respectively. u_{1i} and u_{2j} has a link in G' if and only if v_i is an endpoint of e_j in G. There are another $2|E|$ "branching nodes" in G'. Each u_{1i} is linked with $deg(v_i)$ "branching nodes". We draw the brief structure of G' in Figure 7. Obviously, each u_{1i} has $2deg(v_i)$ neighbors in G', and each u_{2j} has 2 neighbors.

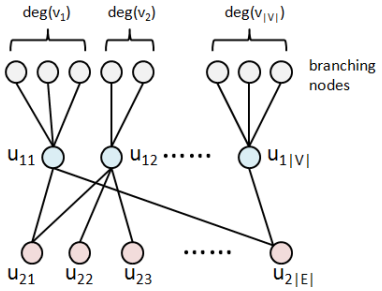


Figure 7: Structure of G' (Case $\delta = \frac{1}{2}$)

We claim that G has a vertex cover of size at most k if and only if G' has a seed set of size at most k to activate the whole graph.

If G has a vertex cover $\{v_{x_1}, v_{x_2}, ..., v_{x_l}\}$ ($l \leq k$), we choose $S_0 = \{u_{1x_1}, u_{1x_2}, ..., u_{1x_l}\}$ as our seed set. According to the property of vertex cover, each u_{2j} has at least 1 active neighbor now, so all u_{2j} are activated after step 1. Then each u_{1i} has at least $deg(v_i)$ active neighbors, so all u_{1i} are activated after step 2. And all "branching nodes" will be activated after step 3. Therefore S_0 activates the whole graph.

If G does not have a vertex cover of size k, we assume that G' has a seed set S_0 of size k to get the contradiction.

(1) If there are any "branching nodes" w in S_0, it will be better if we choose a u_{1i} linked with w instead of w itself, because w will become active soon after u_{1i} is active.

(2) If there are any u_{2j} in S_0, it will be better if we choose a u_{1i} linked with u_{2j} instead of u_{2j} itself, because u_{2j} will become active soon after u_{1i} is active.

According to (1) and (2), we can assume $S_0 = \{u_{1x_1}, u_{1x_2}, ..., u_{1x_l}\}$, $l \leq k$. Since $\{v_{x_1}, v_{x_2}, ..., v_{x_l}\}$ is not a vertex cover of G, there will be at least one u_{2j} remaining inactive after step 1. We suppose that u_{2j} is linked with u_{1s} and u_{1t} in G'. Obviously u_{1s} and u_{1t} are not in S_0, so they are inactive after step 1. Now we get the following "deadlock": If u_{2j} want to be activated, at least one of u_{1s} and u_{1t} should become active first. If u_{1s} or u_{1t} want to be activated, u_{2j} should become active first. So all of these 3 nodes will never become active. We get the contradiction and prove our claim.

Case $\delta \in (0, \frac{1}{2})$. We slightly modify the graph G' in Figure 7. Let each u_{1i} link with K_i "branching nodes" and each u_{2j} link with K_0 "branching nodes", where

$$K_0 = \min\{K \in \mathbb{N} | 0 < \delta \leq \frac{1}{K+2}\},$$
$$K_i = \min\{K \in \mathbb{N} | \frac{deg(u_i)-1}{deg(u_i)+K} < \delta \leq \frac{deg(u_i)}{deg(u_i)+K}\}. \quad (1)$$

Note that the constraint of K_i is equivalent to

$$\frac{deg(u_i)-1}{\delta} - deg(u_i) < K_i \leq \frac{deg(u_i)}{\delta} - deg(u_i), \quad (2)$$

therefore the gap between the upper bound and lower bound is $\frac{1}{\delta} > 1$. So there is at least one integer satisfying the constraint.

Similar to the Case $\delta = \frac{1}{2}$, we can prove that G has a vertex cover of size at most k if and only if G' has a seed set of size at most k to activate the whole graph.

Case $\delta \in (\frac{1}{2}, 1)$. We slightly modify the graph G' in Figure 7. Let each u_{1i} link with K_i "branching nodes" and each u_{2j} link with K_0 "2-node bands", where

$$K_0 = \min\{K \in \mathbb{N} | \frac{K}{K+2} < \delta \leq \frac{K+1}{K+2}\},$$
$$K_i = \min\{K \in \mathbb{N} | \frac{deg(u_i)-1}{deg(u_i)+K} < \delta \leq \frac{deg(u_i)}{deg(u_i)+K}\}. \quad (3)$$

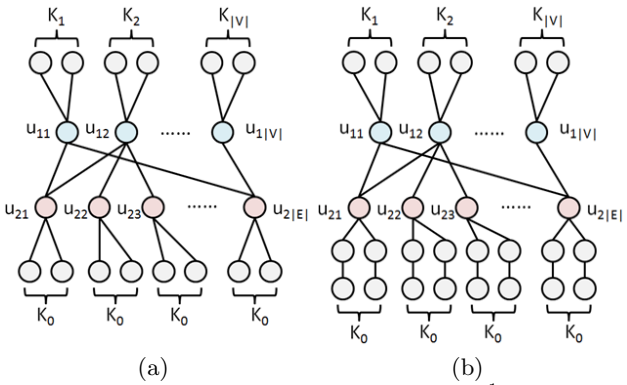
Note that the constraint of K_0 is equivalent to

$$\frac{2\delta-1}{1-\delta} < K_i \leq \frac{2\delta}{1-\delta}, \quad (4)$$

therefore the gap between the upper bound and lower bound is $\frac{1}{1-\delta} > 1$. So there is at least one integer satisfying the constraint. The existence of K_i is the same with Case $\delta \in (0, \frac{1}{2})$.

Similar to the Case $\delta = \frac{1}{2}$, we can prove that G has a vertex cover of size at most k if and only if G' has a seed set of size at most $k + K_0|E|$ to activate the whole graph (since each "2-node band" needs an active seed node).

We draw the brief structures of G' for Case $\delta \in (0, \frac{1}{2})$ and Case $\delta \in (\frac{1}{2}, 1)$ in Figure 8.



(a) (b)

Figure 8: Structure of G'. (a) Case $\delta \in (0, \frac{1}{2})$. (b) Case $\delta \in (\frac{1}{2}, 1)$.

It is easy to see that the construction of G' can be done in polynomial time. So we finish our reduction in the first part. Note that G' in all three cases is a bipartite graph.

The second part is rather easy. We want to minimize $|S_0|$ subject to $|S_n| = |V|$. For $i = 1, 2, ..., |V|$, we solve the influence maximization problem $\max_{|S_0| \leq i} |S_n|$. And the answer to the seed minimization problem is the smallest i

which makes $\max_{|S_0|\leq i} |S_n| = |V|$. $\square$

B. PROOF OF THEOREM 3

PROOF. Our proof is similar to the one in [8] except some modifications. In this proof, we assume that the graph is directed. 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 [34]. 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 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 [8], we have

$$\sigma_{G_i}(S_0) = \sum_{\pi \in \mathcal{P}} \prod_{e \in \pi} w_i, \quad (1 \leq i \leq n+1), \tag{5}$$

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 $\mathcal{P}$ $(0 \leq j \leq n)$. We have

$$\sigma_{G_i}(S_0) = \sum_{j=0}^{n} \sum_{\pi \in B_j} \prod_{e \in \pi} w_i = \sum_{j=0}^{n} \sum_{\pi \in B_j} w_i^j = \sum_{j=0}^{n} w_i^j |B_j|. \tag{6}$$

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. $\square$