

Minimizing Seed Set for Viral Marketing

Cheng Long, Raymond Chi-Wing Wong
 Department of Computer Science and Engineering
 The Hong Kong University of Science and Technology
 {clong, raywong}@cse.ust.hk

Abstract—Viral marketing has attracted considerable concerns in recent years due to its novel idea of leveraging the social network to propagate the awareness of products. Specifically, viral marketing is to first target a limited number of users (seeds) in the social network by providing incentives, and these targeted users would then initiate the process of awareness spread by propagating the information to their friends via their social relationships. Extensive studies have been conducted for maximizing the awareness spread given the number of seeds. However, all of them fail to consider the common scenario of viral marketing where companies hope to use as few seeds as possible yet influencing at least a certain number of users. In this paper, we propose a new problem, called *J-MIN-Seed*, whose objective is to minimize the number of seeds while at least J users are influenced. *J-MIN-Seed*, unfortunately, is proved to be NP-hard in this work. In such case, we develop a greedy algorithm that can provide error guarantees for *J-MIN-Seed*. Furthermore, for the problem setting where J is equal to the number of all users in the social network, denoted by *Full-Coverage*, we design other efficient algorithms. Extensive experiments were conducted on real datasets to verify our algorithm.

I. INTRODUCTION

Viral marketing is an advertising strategy that takes the advantage of the effect of “word-of-mouth” among the relationships of individuals to promote a product. Instead of covering *massive* users directly as traditional advertising methods [1] do, viral marketing targets a *limited* number of *initial* users (by providing *incentives*) and utilizes their social relationships, such as friends, families and co-workers, to further spread the awareness of the product among individuals. Each individual who gets the awareness of the product is said to be *influenced*. The number of all influenced individuals corresponds to the *influence* incurred by the initial users. According to some recent research studies [2], people tend to trust the information from their friends, relatives or families more than that from general advertising media like TVs. Hence, it is believed that viral marketing is one of the most effective marketing strategies [3]. In fact, extensive commercial instances of viral marketing succeed in real life. For example, *Nike Inc.* used social networking websites such as *orkut.com* and *facebook.com* to market products successfully [4].

The propagation process of viral marketing within a social network can be described in the following way. At the beginning, the advertiser selects a set of initial users and provides these users incentives so that they are willing to initiate the awareness spread of the product in the social network. We call these initial users *seeds*. Once the propagation is initiated, the information of the product *diffuses* or *spreads*

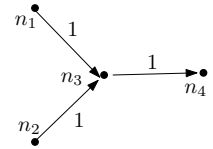
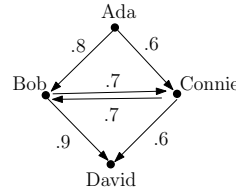


Fig. 1. Social network (IC model) Fig. 2. Counter example ($\alpha(\cdot)$)

via the relationships among users in the social network. A lot of models about how the above diffusion process works have been proposed [5-10]. Among them, the *Independent Cascade Model (IC model)* [5, 6] and the *Linear Threshold Model (LT model)* [7, 8] are the two that are widely used in the literature. In the social network, the IC model simulates the situation where for each influenced user u , each of its neighbors has a probability to be influenced by u , while the LT model captures the phenomenon where each user’s tendency to become influenced increases when more of its neighbors become influenced.

Consider the following scenario of viral marketing. A company wants to advertise a new product via viral marketing within a social network. Specifically, it hopes that at least a certain number of users, says J , in the social network must be influenced yet the number of seeds for viral marketing should be as small as possible. Clearly, the above problem can be formalized as follows. Given a social network $G(V, E)$, we want to find a set of seeds such that the size of the seed set is minimized and at least J users are influenced at the end of viral marketing. We call this problem *J-MIN-Seed*.

We use Figure 1 to illustrate the main idea of the *J-MIN-Seed* problem. The four nodes shown in Figure 1 represent four members in a family, namely Ada, Bob, Connie and David, respectively. In the following, we use the terms “nodes” and “users” interchangeably since they correspond to the same concept. The directed edge (u, v) with the weight of $w_{u,v}$ indicates that node u has the probability of $w_{u,v}$ to influence node v for the awareness of the product. Now, we want to find the *smallest* seed set such that *at least 3* nodes can be influenced by this seed set. It is easy to verify that the expected influence incurred by seed set $\{Ada\}$ is about 3.57^1 under the IC model and no smaller seed set can incur at least 3 influenced nodes. Hence, seed set $\{Ada\}$ is our solution.

J-MIN-Seed can be applied to most (if not all) applica-

¹The computation of the expected influence incurred by a seed is calculated by considering all cascades from this seed. E.g., the expected influence on Bob incurred by Ada is $1 - (1 - 0.8) \cdot (1 - 0.6 \cdot 0.7) = 0.884$.

tions of viral marketing. Intuitively, J -MIN-Seed asks for the minimum cost (seeds) while satisfying an explicit requirement of revenue (influenced nodes). Clearly, in the mechanism of viral marketing, a seed and an influenced node correspond to cost and potential revenue of a company, respectively. Because the company has to *pay* the seeds for incentives, while an influenced node might bring revenue to the company. In many cases, companies face the situation where the goal of revenue has been set up explicitly and the cost should be minimized. Thus, J -MIN-Seed meets these companies' demands.

Another area where J -MIN-Seed can be widely used is the "majority-decision rule" (e.g., the three-fifths majority rule in the US Senate). By majority-decision rule, we mean the principle under which the decision is determined by the majority (or a certain portion) of participants. That is, in order to affect a group of people to make a decision, e.g., purchasing our products, we *only* need to convince a certain number of members in this group, says J , which is the threshold of the number of people to agree on the decision. Clearly, for these kinds of applications, J -MIN-Seed could be used to affect the decision of the whole group and yield the minimum cost. In fact, J -MIN-Seed is particularly useful in the election campaigns where the "majority-decision rule" is adopted.

No existing studies have been conducted for J -MIN-Seed even though it plays an essential role in the viral marketing field. In fact, most existing studies related to viral marketing focus on maximizing the influence incurred by a certain number of seeds, says k [11-16]. Specifically, they aim at maximizing the number of influenced nodes when only k seeds are available. We denote this problem by k -MAX-Influence. Clearly, J -MIN-Seed and k -MAX-Influence have different goals with different given resources.

Naïvely, we can solve the J -MIN-Seed problem by adapting an existing algorithm for k -MAX-Influence. Let k be the number of seeds. We set $k = 1$ at the beginning and increment k by 1 at the end of each iteration. For each iteration, we use an existing algorithm for k -MAX-Influence to calculate the maximum number of nodes, denoted by I , that can be influenced by a seed set with the size equal to k . If $I \geq J$, we stop our process and return the current number k . Otherwise, we increment k by 1 and perform the next iteration. However, this naïve method is very time-consuming since it issues the existing algorithm for k -MAX-Influence many times for solving J -MIN-Seed. Note that k -MAX-Influence is NP-hard [12]. Any existing algorithm for k -MAX-Influence is computation-expensive, which results in this naïve method with a high computation cost. Hence, we should resort to other more efficient solutions.

In this paper, J -MIN-Seed is, unfortunately, proved to be NP-hard. Motivated by this, we design an approximate (greedy) algorithm for J -MIN-Seed. Specifically, our algorithm iteratively adds into a seed set one node that generates the greatest influence gain until the influence incurred by the seed set is at least J . Besides, we work out an additive error bound and a multiplicative error bound for this greedy algorithm.

In some cases, the companies would set the parameter J for J -MIN-Seed to be the *total* number of users in the underlying social network since they want to influence as many users as possible. Motivated by this, we further discuss our J -MIN-Seed problem under the special setting where $J = |V|$ (the total number of users). We call this special instance of J -MIN-Seed as *Full-Coverage* for which we design other efficient algorithms.

We summarize our contributions as follows. Firstly, to the best of our knowledge, we are the first to propose the J -MIN-Seed problem, which is a fundamental problem in viral marketing. Secondly, we prove that J -MIN-Seed is NP-hard in this paper. Under such situation, we develop a greedy algorithm framework for J -MIN-Seed, which, fortunately, can provide error guarantees for the approximation error. Thirdly, for the Full-Coverage problem (i.e., J -MIN-Seed where $J = |V|$), we observe some interesting properties and thus design some other efficient algorithms. Finally, we conducted extensive experiments which verified our algorithms.

The rest of the paper is organized as follows. Section II covers the related work of our problem, while Section III provides the formal definition of the J -MIN-Seed problem and some relevant properties. We show how to calculate the influence incurred by a seed set in Section IV, which is followed by Section V discussing our greedy algorithm framework. In Section VI, we discuss the Full-Coverage problem. We conduct our empirical studies in Section VII and conclude our paper in Section VIII.

II. RELATED WORK

In Section II-A, we discuss two widely used *diffusion models* in a social network, and in Section II-B, we give the related work about the *influence maximization* problem.

A. Diffusion Models

Given a social network represented in a directed graph G , we denote V to be the set containing all the nodes in G each of which corresponds to a user and E to be the set containing all the directed edges in G . Each edge $e \in E$ in form of (u, v) is associated with a weight $w_{u,v} \in [0, 1]$. Different diffusion models have different meanings on weights. In the following, we discuss the meanings for two popular diffusion models, namely the Independent Cascade (IC) model and the Linear Threshold (LT) model.

1) *Independent Cascade (IC) Model* [7, 8]: The first model is the Independent Cascade (IC) model. In this model, the influence is based on how a single node influences each of its *single* neighbor. The weight $w_{u,v}$ of an edge (u, v) corresponds to the probability that node u influences node v . Let S_0 be the initial set of influenced nodes (seeds in our problem). The diffusion process involves a number of *steps* where each step corresponds to the influence spread from some influenced nodes to other non-influenced nodes. At step t , all influenced nodes at step $t - 1$ remain influenced, and each node that becomes influenced at step $t - 1$ for the *first* time has one chance to influence its non-influenced neighbors.

Specifically, when an influenced node u attempts to influence its non-influenced neighbor v , the probability that v becomes influenced is equal to $w_{u,v}$. The propagation process halts at step t if no nodes become influenced at step $t-1$. The running example in Figure 1 is based on the IC model.

For a graph under the IC model, we say that the graph is *deterministic* if all its edges have the probabilities equal to 1. Otherwise, we say that it is *probabilistic*.

2) *Linear Threshold (LT) Model* [5, 6]: The second model is the Linear Threshold (LT) model. In this model, the influence is based on how a single node is influenced by its *multiple* neighbors together. The weight $w_{u,v}$ of an edge (u, v) corresponds to the relative strength that node v is influenced by its neighbor u (among all of v 's neighbors). Besides, for each $v \in V$, it holds that $\sum_{(u,v) \in E} w_{u,v} \leq 1$. The dynamics of the process proceeds as follows. Each node v selects a threshold value θ_v from range $[0, 1]$ randomly. Same as the IC model, let S_0 be the set of initial influenced nodes. At step t , the non-influenced node v , for which the total weight of the edges from its *influenced* neighbors exceeds its threshold ($\sum_{(u,v) \in E \text{ and } u \text{ is influenced}} w_{u,v} \geq \theta_v$), becomes influenced. The spread process terminates when no more influence spread is possible.

For a graph under the LT model, we say that the graph is *deterministic* if the thresholds of all its nodes have been set before the process of influence spread. Otherwise, we say that it is *probabilistic*.

B. Influence Maximization

Motivated by the fact that social network plays a fundamental role in spreading ideas, innovations and information, Domingoes and Richardson proposed to use social networks for marketing purpose, which is called viral marketing [11, 17]. By viral marketing, they aimed at selecting a limited number of seeds such that the influence incurred by these seeds is maximized. We call this fundamental problem as the *influence maximization* problem.

In [12], Kempe et al. formalized the above influence maximization problem as a discrete optimization problem called *k-MAX-Influence* for the first time. *Given a social network $G(V, E)$ and an integer k , find k seeds such that the incurred influence is maximized.* Kempe et al. proved that *k-MAX-Influence* is NP-hard for both the IC model and the LT model. To achieve better efficiency, they provided a $(1 - 1/e)$ -approximation algorithm for *k-MAX-Influence*.

Recently, several studies have been conducted to solve *k-MAX-Influence* in a more efficient and/or scalable way than the aforementioned approximate algorithm in [12]. Specifically, in [13], Leskovec et al. employed a ‘‘lazy-forward’’ strategy to select seeds, which has been shown to be effective for reducing the cost of the influence propagation of nodes. In [14], Kimura et al. proposed a new shortest-path cascade model, based on which, they developed efficient algorithms for *k-MAX-Influence*. Motivated by the drawback of non-scalability of all aforementioned solutions for *k-MAX-Influence*, Chen et al. proposed an arborescence-based heuris-

tic algorithm, which was verified to be quite scalable to large-scale social networks [15].

The influence maximization problem has been extended into the setting with multiple products instead of a single product. Bharathi et al. solved the influence maximization problem for multiple *competitive* products using game-theoretical methods [18, 19], while Datta et al. proposed the influence maximization problem for multiple *non-competitive* products [16]. Apart from these studies aiming at maximizing the influence, considerable efforts have been devoted to the diffusion models in social networks [9, 10].

Clearly, most of the existing studies related to viral marketing aim at maximizing the influence incurred by a limited number of seeds (i.e., *k-MAX-Influence*). While our problem, *J-MIN-Seed*, is targeted to minimize the number of seeds while satisfying the requirement of influencing at least a certain number of users in the social network. As discussed in Section I, a naïve adaption of any existing algorithm for *k-MAX-Influence* is time-consuming.

III. PROBLEM

We first formalize *J-MIN-Seed* in Section III-A. In Section III-B, we provide several properties related to *J-MIN-Seed*.

A. Problem Definition

Given a set S of seeds, we define the *influence* incurred by the seed set S (or simply the influence of S), denoted by $\sigma(S)$, to be the expected number of nodes influenced during the diffusion process initiated by S . How to calculate $\sigma(S)$ under different diffusion models given S will be discussed in Section IV.

Problem 1 (J-MIN-Seed): Given a social network $G(V, E)$ and an integer J , find a set S of seeds such that the size of the seed set is minimized and $\sigma(S) \geq J$. ■

We say that node u is *covered* by seed set S if u is influenced during the influence diffusion process initiated by S . It is easy to see that *J-MIN-Seed* aims at minimizing the number of seeds while satisfying the requirement of covering at least J nodes. Given a node x in V and a subset S of V , the *marginal gain* of inserting x into S , denoted by $G_x(S)$, is defined to be $\sigma(S \cup \{x\}) - \sigma(S)$.

We show the hardness of *J-MIN-Seed* with the following theorem.

Theorem 1: The *J-MIN-Seed* problem is NP-hard for both the IC model and the LT model. ■

B. Properties

Since the analysis of the error bounds of our approximate algorithms to be discussed is based on the property that function $\sigma(\cdot)$ is *submodular*, we first briefly introduce the concept of submodular function, denoted by $f(\cdot)$. After that, we provide several properties related to the influence diffusion process in a social network.

Definition 1 (Submodularity): Let U be a universe set of elements and S be a subset of U . Function $f(\cdot)$ which maps

S to a non-negative value is said to be *submodular* if given any $S \subseteq U$ and any $T \subseteq U$ where $S \subseteq T$, it holds for any element $x \in U - T$ that $f(S \cup \{x\}) - f(S) \geq f(T \cup \{x\}) - f(T)$. ■

In other words, we say $f(\cdot)$ is *submodular* if it satisfies the “diminishing marginal gain” property: the marginal gain of inserting a new element into a set T is at most the marginal gain of inserting the same element into a subset of T .

According to [12], function $\sigma(\cdot)$ is submodular for both the IC model and the LT model. The main idea is as follows. When we add a new node x into a seed set S , the influence incurred by the node x (without considering the nodes in S) might overlap with that incurred by S . The larger S is, the more overlap might happen. Hence, the marginal gain is smaller on a (larger) set compared to that on any of its subsets. We formalize this statement with the following Property 1. The proof can be found in [12].

Property 1: Function $\sigma(\cdot)$ is submodular for both the IC model and the LT model. ■

To illustrate the concept of submodular functions, consider Figure 1. Assume that a seed set T is $\{Ada\}$. Let a subset S of T be \emptyset . We insert into seed sets T and S the same node Bob . In fact, it is easy to calculate $\sigma(\emptyset) = 0$, $\sigma(\{Ada\}) = 3.57$, $\sigma(\{Bob\}) = 2.64$ and $\sigma(\{Ada, Bob\}) = 3.83$. Consequently, we know the *marginal gain* of adding a new node Bob into set T , i.e., $\sigma(\{Ada, Bob\}) - \sigma(\{Ada\}) = 0.26$, is smaller than that of adding Bob into one of its subsets S , i.e., $\sigma(\{Bob\}) - \sigma(\emptyset) = 2.64$.

In the k -MAX-Influence problem, we have a submodular function $\sigma(\cdot)$ which takes a set of seeds as an input and returns the expected number of influenced nodes incurred by the seed set as an output. Similarly, in the J -MIN-Seed problem, we define a function $\alpha(\cdot)$ which takes a set of influenced nodes as an input and returns the smallest number of seeds needed to influence these nodes as an output. One may ask: *Is function $\alpha(\cdot)$ also submodular?* Unfortunately, the answer is “no” which is formalized with the following Property 2.

Property 2: Function $\alpha(\cdot)$ is *not* submodular for both the IC model and the LT model. ■

Proof. We prove Property 2 by constructing a problem instance where $\alpha(\cdot)$ does not satisfy the aforementioned conditions of a submodular function. We first discuss the case for the IC model. Consider the example as shown in Figure 2. In this figure, there are four nodes, namely n_1 , n_2 , n_3 and n_4 . We assume that each edge is associated with its weight equal to 1, which indicates that an influenced node u will influence a non-influenced node v *definitely* when there is an edge from u to v . Let set T be $\{n_1, n_3, n_4\}$ and a subset of T , says S , be $\{n_3, n_4\}$. Obviously, when node n_1 is influenced, it will further influence node n_3 and node n_4 , i.e., all the nodes in T will be influenced when n_1 is selected as a seed. Thus, $\alpha(T) = 1$. Similarly, we know that $\alpha(S) = 1$. Now, we add node n_2 into both T and S and then obtain $\alpha(T \cup \{n_2\}) = 2$ (by the seed set $\{n_1, n_2\}$) and $\alpha(S \cup \{n_2\}) = 1$ (by the seed set $\{n_2\}$). As a result, we know that $\alpha(T \cup \{n_2\}) - \alpha(T) = 1 > \alpha(S \cup \{n_2\}) - \alpha(S) = 0$, which, however, violates the conditions of a submodular function.

Next, we discuss the case for the LT model. Consider the special case where each node’s threshold is equal to a value slightly greater than 0. Consequently, a node will be influenced whenever one of its neighbors becomes influenced. The resulting diffusion process is actually identical to the special case for the IC model where the weights of all edges are 1s. That is, the example in Figure 2 can also be applied for the LT model. Hence, Property 2 also holds for the LT model. ■

Property 2 suggests that we cannot directly adapt existing techniques for the k -MAX-Influence problem (which involves a *submodular* function as an objective function) to our J -MIN-Seed problem (which involves a *non-submodular* function as an objective function).

IV. INFLUENCE CALCULATION

We describe how we compute the influence of a given seed set (i.e., $\sigma(\cdot)$) under the IC model (Section IV-A) and the LT model (Section IV-B).

A. IC model

It has been proved in [15] that the process of calculating the influence given a seed set for the IC model is *#P-hard*. That is, computing the *exact* influence is hard. Thus, we have to resort to approximate algorithms for efficiency.

Intuitively, the hardness of calculating the influence is due to the fact that the edges in the social network under the IC model are *probabilistic* in the sense that the propagation of influence via an edge happens with probability. In contrast, when the social network is *deterministic*, i.e., the probability associated with each edge is exactly 1, we only need to *traverse* the graph from each seed in a breadth-first manner and return all visited nodes as the influenced nodes incurred by the seed set, thus resulting in a linear-time algorithm for influence calculation.

In view of the above discussion, we use *sampling* to calculate the (approximate) influence as follows. Let $G(V, E)$ be the original probabilistic social network and S be the seed set. Instead of calculating the influence on G directly, we calculate the influence on each *sampled graph* from G using the same seed set S and finally average the incurred influences on all sampled graphs to obtain the approximate one for the original probabilistic graph. To obtain the sampled graph of $G(V, E)$ each time, we keep the node set V unchanged, remove the edge (u, v) with the probability of $1 - w_{u,v}$ for each edge $(u, v) \in E$ and assign each remaining edge with the weight equal to 1. In this way, we can obtain that the probability that an edge (u, v) remains in the resulting graph is $w_{u,v}$. Note that the resulting sampled graph is *deterministic*. We call such a process as *social network sampling*.

Conceptually, given a probabilistic graph $G(V, E)$, each G ’s sampled graph is generated with probability equal to a certain value. As a result, the influence calculated based on each G ’s sampled graph has one specific probability to be equal to the exact influence on the original probabilistic graph G given the same seed set. That is, the *exact influence* for G is the *expected influence* for a sampled graph of G . Based on this,

we can use *Hoeffding's Inequality* to analyze the error incurred by our sampling method. We state our result with the following Lemma 1.

Lemma 1: Let c be a real number between 0 and 1. Given a seed set S and a social network $G(V, E)$ under the IC model, the sampling method stated above achieves a $(1 \pm \epsilon)$ -approximation of the influence incurred by S on G with the confidence at least c by performing the sampling process at least $\frac{(|V|-1)^2 \ln(2/(1-c))}{2\epsilon^2 |S|^2}$ times. ■

B. LT model

Similar to the case under the IC model, the influence calculation for the LT model is much easier when the graph is *deterministic* (i.e., the threshold of each node has been specified before the process of influence spread). We illustrate the main idea as follows. For an influenced node u , all we need to do is to add the corresponding influence to each of its non-influenced neighbors and check whether each of its non-influenced neighbors, says v , has received enough influence (θ_v) to be influenced. If so, we change node v to be influenced. Otherwise, we leave node v non-influenced. At the beginning, we initialize the set of influenced nodes to be the seed set S . Then, we perform the above process for each influenced node until no new influenced nodes are generated.

With the view of the above discussion, we perform the influence calculation on a probabilistic graph for the LT model in the same way as the IC model except for the sampling method. Specifically, to sample a probabilistic graph G under the LT model, we pick a real number from range $[0, 1]$ uniformly as the threshold of each node in G to form a deterministic graph. We perform the sampling process multiple times, and for each resulting deterministic graph, we run the algorithm for a deterministic graph (just described above) to obtain the incurred influence. Finally, we average the influences on all sampled graphs to obtain the approximate influence.

Clearly, we can derive a similar lemma as Lemma 1 for the LT model.

V. GREEDY ALGORITHM

We present in Section V-A the framework of our greedy algorithm which finds a seed set by adding a seed into the seed set iteratively. Section V-B provides the analysis of this algorithm framework, while Section V-C discusses two different implementations of this algorithm framework.

A. Algorithm Framework

As proved in Section III, J -MIN-Seed is NP-hard. It is expected that there is no efficient exact algorithm for J -MIN-Seed. As discussed in Section I, if we want to solve J -MIN-Seed, a naïve adaption of any existing algorithm originally designed for k -MAX-Influence is time-consuming. The major reason is that it executes an existing algorithm many times and the execution of this existing algorithm for an iteration is *independent* of the execution of the same algorithm for the next iteration. Motivated by this observation, we propose a greedy algorithm which solves J -MIN-Seed efficiently by

Algorithm 1 Greedy Algorithm Framework

Input: $G(V, E)$: a social network.

J : the required number of nodes to be influenced

Output: S : a seed set.

```

1:  $S \leftarrow \emptyset$ 
2: while  $\sigma(S) < J$  do
3:    $u \leftarrow \arg \max_{x \in V-S} (\sigma(S \cup \{x\}) - \sigma(S))$ 
4:    $S \leftarrow S \cup \{u\}$ 
5: return  $S$ 

```

executing an iteration based on the results from its previous iteration.

Specifically, we first initialize a seed set S to be an empty set. Then, we select a non-seed node u such that the marginal gain of inserting u into S is the greatest and then we insert u into S . We repeat the above steps until at least J nodes are influenced. Algorithm 1 presents this greedy algorithm framework.

This greedy algorithm is similar to the algorithm from [12] for k -MAX-Influence except the stopping criterion, but they have different theoretical results. The stopping criterion in this greedy algorithm is $\sigma(S) \geq J$ and the stopping criterion in the algorithm from [12] is $|S| \geq k$ where k is a user parameter of k -MAX-Influence. Note that our greedy algorithm for J -MIN-Seed has theoretical results which guarantee the number of *seeds* used while the algorithm for k -MAX-Influence has theoretical results which guarantee the number of *influenced nodes*.

B. Theoretical Analysis

In this part, we show that the greedy algorithm framework in Algorithm 1 can return the seed set with both an additive error guarantee and a multiplicative error guarantee.

The greedy algorithm gives the following additive error bound.

Lemma 2 (Additive Error Guarantee): Let h be the size of the seed set returned by the greedy algorithm framework in Algorithm 1 and t be the size of the optimal seed set for J -MIN-Seed. The greedy algorithm framework in Algorithm 1 gives an additive error bound equal to $1/e \cdot J + 1$. That is, $h - t \leq 1/e \cdot J + 1$. Here, e is the natural logarithmic base. ■

Before we give the multiplicative error bound of the greedy algorithm, we first give some notations. Suppose that the greedy algorithm terminates after h iterations. We denote S_i to be the seed set maintained by the greedy algorithm at the end of iteration i where $i = 1, 2, \dots, h$. Let S_0 denote the seed set maintained before the greedy algorithm starts (i.e., an empty set). Note that $\sigma(S_i) < J$ for $i = 1, 2, \dots, h - 1$ and $\sigma(S_h) \geq J$.

In the following, we give the multiplicative error bound of the greedy algorithm framework in Algorithm 1.

Lemma 3 (Multiplicative Error Guarantee): Let $\sigma'(S) = \min\{\sigma(S), J\}$. The greedy algorithm framework in Algorithm 1 is a $(1 + \min\{k_1, k_2, k_3\})$ -approximation of J -MIN-Seed, where $k_1 = \ln \frac{J}{J - \sigma'(S_{h-1})}$, $k_2 = \ln \frac{\sigma'(S_1)}{\sigma'(S_h) - \sigma'(S_{h-1})}$, and

$$k_3 = \ln(\max\{\frac{\sigma'(\{x\})}{\sigma'(S_i \cup \{x\}) - \sigma'(S_i)} | x \in V, 0 \leq i \leq h, \sigma'(S_i \cup \{x\}) - \sigma'(S_i) > 0\}). \quad \blacksquare$$

C. Implementations

As can be seen, the efficiency of Algorithm 1 relies on the calculation of the influence of a given seed set (operator $\sigma(\cdot)$). However, the influence calculation process for the IC model is #P-hard [15]. Under such a circumstance, we adopt the sampling method discussed in Section IV when using operator $\sigma(\cdot)$. We denote this implementation by *Greedy1*.

In fact, we have an alternative implementation of Algorithm 1 as follows. Instead of sampling the social network to be deterministic *when* calculating the influence incurred by a given seed set, we can sample the social network to generate a certain number of deterministic graphs *only* at the beginning. Then, we solve the J -MIN-Seed problem on each such deterministic graph using Algorithm 1, where the cost of operator $\sigma(\cdot)$ simply becomes the time to traverse the graph.

At the end, we return the average of the sizes of the seed sets returned by the algorithm based on all samples (deterministic graphs). We call this alternative implementation as *Greedy2*.

VI. FULL-COVERAGE

In some applications, we are interested in influencing all nodes in the social network. For example, a government wants to promote some campaigns like an election and an awareness of some infectious diseases. In these applications, J is set to $|V|$. We call this special instance of J -MIN-Seed as *Full-Coverage*. In Section VI-A, we give some interesting observations and present an efficient algorithm on deterministic graphs for the IC model, while in Section VI-B, we develop our probabilistic algorithm which can provide an arbitrarily small error for the IC model.

A. Full-Coverage on Deterministic Graph (IC Model)

According to Theorem 1, *in general*, it is NP-hard to solve the J -MIN-Seed problem on a graph (either probabilistic or deterministic) for the IC model. However, on a *deterministic* graph for the IC model, Full-Coverage is not NP-hard yet easy to solve. In the following, we design an efficient algorithm to handle Full-Coverage on a deterministic graph $G(V, E)$.

Before illustrating our efficient method for Full-Coverage, we first introduce the following two observations.

Observation 1: On a deterministic graph, if a node within a strongly connected component (SCC) is influenced, then it will influence *all* nodes in this SCC. \blacksquare

Observation 2: Any node with in-degree equal to 0 must be selected as a seed in order to be influenced. This is because it cannot be influenced by other nodes. \blacksquare

Based on the above two observations, we design our method called *Decompose-and-Pick* as follows.

At the first step, we decompose the deterministic graph into a number of strongly connected components (SCCs), namely $scc_1, scc_2, \dots, scc_m$. This step can be achieved by adopting some existing methods in the rich literature for finding all SCCs in a graph. In our implementation for this step, we adopt

the SCC computation algorithm developed by Kosaraju et al [20], which runs in $O(|V| + |E|)$ time.

For the second step, we construct a new graph $G(V', E')$ based on $G(V, E)$. Specifically, for constructing V' , we create a new node v_i for each SCC scc_i obtained in Step 1. We construct E' as follows. Initially, E' is set to an empty set. For each $(u, v) \in E$, we find the SCC containing u (v) in $G(V, E)$, says scc_i (scc_j). Then, we find the node v_i (v_j) representing scc_i (scc_j) in $G(V', E')$. We check whether $(v_i, v_j) \in E'$. If not, we insert (v_i, v_j) into E' . Clearly, the cost for constructing V' is $O(|V|)$, while the cost for generating E' is $O(|E| \cdot C_{check})$, where C_{check} indicates the cost for checking whether a specific edge (v_i, v_j) has been constructed before in E' . C_{check} depends on the structure for storing $G(V', E')$. Specifically, C_{check} is $O(1)$ when $G(V', E')$ is stored in an *adjacency matrix*. With this data structure, the overall cost for Step 2 is $O(|V| + |E|)$. In case that $G(V', E')$ is maintained in an *adjacency list*, C_{check} becomes $O(|E'|)$ (bounded by $O(|E|)$), resulting in Step 2's complexity equal to $O(|V| + |E|^2)$ in the worst case. To further reduce the complexity of Step 2 in this case, we do not check the existence of each newly formed edge in the new graph every time we create a new edge. Instead, we create all newly formed edges and perform sorting on all the newly formed edges to filter out any redundant edges in E' , thus yielding the cost of Step 2 equal to $O(|V| + |E| \cdot \log |E|)$. Note that there exist no SCCs in the constructed graph $G'(V', E')$.

For the last step, we simply pick the nodes with in-degree equal to 0 in $G(V', E')$ and for each such node v_i , we insert into the seed set S a node randomly from its corresponding scc_i in the original $G(V, E)$. Since there exist no SCCs in $G'(V', E')$, it is possible to perform a *topological sort* on $G'(V', E')$. Hence, the seed set consisting of all the nodes with in-degree equal to 0 in $G'(V', E')$ would influence all nodes in $G'(V', E')$. Since each node in $G'(V', E')$ corresponds to a SCC structure in $G(V, E)$, according to Observation 2, we conclude that the seed set S constructed at the last step would influence $|V|$ nodes in $G(V, E)$ (deterministic). Clearly, the cost of Step 3 is $O(|V'| + |E'|)$ (DFS/BFS), which is bounded by $O(|V| + |E|)$.

In summary, the worst-case time complexity of Decompose-and-Pick is $O(|V| + |E|)$ and $O(|V| + |E| \cdot \log |E|)$ when the new graph is maintained in an adjacency matrix and an adjacency list, respectively.

B. Full-Coverage on Probabilistic Graph (IC Model)

At this moment, it is quite straightforward to perform our probabilistic algorithm for Full-Coverage based on *social network sampling* and *Decompose-and-Pick* as follows. We first use social network sampling to generate a certain number of deterministic graphs. Then, on each such deterministic graph, we run Decompose-and-Pick to obtain its corresponding seed set which covers all the nodes in the social network and the corresponding size of the seed set. At the end, we average the sizes of the seed sets obtained for all samples (deterministic graphs) to approximate the solution of Full-Coverage on a

general (probabilistic) social network. Again, using *Hoeffding's Inequality*, for a real number c between 0 and 1, we can provide users with a $(1 \pm \epsilon)$ -approximation solution for any positive real number ϵ with confidence at least c by performing the sampling process at least $\frac{(|V|-1)^2 \ln(2/(1-c))}{2\epsilon^2}$ times. The proof is similar to that of Lemma 1.

VII. EMPIRICAL STUDY

We set up our experiments in Section VII-A and give the corresponding experimental results in Section VII-B.

A. Experimental Setup

We conducted our experiments on a 2.26GHz machine with 4GB memory under a Linux platform. All algorithms were implemented in C/C++.

1) *Datasets*: We used four real datasets for our empirical study, namely *HEP-T*, *Epinions*, *Amazon* and *DBLP*. *HEP-T* is a collaboration network generated from “High Energy Physics-Theory” section of the e-print arXiv (<http://www.arXiv.org>). In this collaboration network, each node represents one specific author and each edge indicates a co-author relationship between the two authors corresponding to the nodes incident to the edge. The second one, *Epinions*, is a *who-trust-whom* network at *Epinions.com*, where each node represents a member of the site and the link from member u to member v means that u trusts v (i.e., v has a certain influence on u). The third real dataset, *Amazon*, is a product co-purchasing network extracted from *Amazon.com* with nodes and edges representing products and co-purchasing relationships, respectively. We believe that product u has an influence on product v if v is purchased often with u . Both of *Epinions* and *Amazon* are maintained by Jure Leskovec. Our last real dataset, *DBLP*, is another collaboration network of computer science bibliography database maintained by Michael Ley. We summarize the features of the above real datasets in Table I. For efficiency, we ran our algorithms on the samples of the aforementioned real datasets with the sampling ratio equal to one percent. The sampling process is done as follows. We randomly choose a node as the root and then perform a breadth-first traversal (BFT) from this root. If the BFT from one root cannot cover our targeted number of nodes, we continue to pick more new roots randomly and perform BFTs from them until we obtain our expected number of nodes. Next, we construct the edges by keeping the original edges between the nodes traversed.

2) *Configurations*: (1) *Weight generation for the IC model*: We use the QUADRIVALENCY model to generate the weights. Specifically, for each edge, we uniformly choose a value from set $\{0.1, 0.25, 0.5, 0.75\}$, each of which represents minor, low, medium and high influence, respectively. (2) *Weight generation for the LT model*: For each node u , let d_u denote its in-degree, we assign the weight of each edge to u as $1/d_u$. In this case, each node obtains the equivalent influence from each of its neighbors. (3) *No. of Times for Sampling*: For each influence calculation under both the IC model and the LT model, we perform the graph sampling process 10000 times by default. (4) *Parameter J* : In the following, we denote

TABLE I
STATISTICS OF REAL DATASETS

Dataset	HEP-T	Epinions	Amazon	DBLP
No. of Nodes	15233	75888	262111	654628
No. of Edges	58891	508837	1234877	1990259

parameter J as a *relative* real number between 0 and 1 denoting the fraction of the influenced nodes among all nodes in the social network (instead of an *absolute* positive integer denoting the total number of influenced nodes) because a relative measure is more meaningful than an absolute measure in the experiments. We set J to be 0.5 by default. Alternative configurations considered are $\{0.1, 0.25, 0.5, 0.75, 1\}$.

3) *Algorithms*: We compare our greedy algorithm with several other common heuristic algorithms. We list all the algorithms studied in our experiments as follows. (1) *Greedy1*: We denote our first implementation of Algorithm 1 by Greedy1. As stated before, we *only* conduct the *graph sampling process* when performing the influence calculation. (2) *Greedy2*: Greedy2 corresponds to the alternative implementation of Algorithm 1. (3) *Degree-heuristic*: We implemented this baseline algorithm using the heuristic of nodes’ out-degree. Specifically, we repeatedly pick the node with the largest out-degree yet un-covered and add it into the seed set until the incurred influence exceeds the threshold. We denote this heuristic algorithm as *Degree-heuristic*. (4) *Centrality-heuristic*: *Centrality-heuristic* indicates another heuristic algorithm based on the nodes’ *distance centrality*. In sociology, distance centrality is a common measurement of nodes’ importance in a social network based on the assumption that a node with short distances to other nodes would probably have a higher chance to influence them. In *Centrality-heuristic*, we select the seeds in a decreasing order of nodes’ distance centralities until the requirement of influencing at least J nodes is met. (5) *Random*: Finally, we consider the method of selecting seeds from the un-covered nodes at random as a baseline. Correspondingly, we denote it by *Random*.

In the experiment, we do not compare our algorithms with the naïve adaption of an existing algorithm for k -MAX-Influence described in Section I because this naïve adaption is time-consuming as discussed in Section V.

B. Experiment Results

1) *No. of Seeds*: We measure the quality of the algorithm for J -MIN-Seed by using the number of seeds returned by the algorithm. Clearly, the fewer the seeds an algorithm returns, the better it is.

For the IC model, we study the qualities of the five aforementioned algorithms by comparing the number of seeds returned by them. Specifically, we vary parameter J from 0.1 to 1. The experimental results are shown in Figure 3. Consider the results on *HEP-T* (Figure 3(a)) as an example. We find algorithms Greedy1 and Greedy 2 are comparable in terms of quality. Both of them outperform other heuristic algorithms significantly. Similar results can be found in other real datasets.

For the LT model, we conducted the similar experiments, whose results are shown in Figure 6. Same as the IC model,

Greedy1 and Greedy2 beat other algorithms by an order of almost one magnitude in terms the number of returned seeds.

2) *Running Time*: We explore the efficiency of different algorithms by comparing their running times. Again, we vary J , and for each setting of J , we record the corresponding running time of each algorithm.

For the IC model, according to the results shown in Figure 4, we find that Greedy1 is the slowest algorithm. The reason is that Greedy1 selects the seeds by calculating the marginal gain of each non-seed at each iteration and then picking the one with the largest marginal gain while other heuristic algorithms simply choose the non-seed with the best heuristic value (e.g., out-degree and centrality). However, the alternative implementation of our greedy algorithm, i.e., Greedy2, shows its advantage in terms of efficiency. Greedy2 is faster than Greedy1 because the total cost of sampling in Greedy2 is much smaller than that in Greedy1. Besides, Random is slower than Greedy2, though the cost of choosing a seed in Random is $O(1)$. This is because Random usually has to select more seeds than Greedy2 in order to incur the same amount of influence and for each iteration, Random also needs to calculate the influence incurred by the current seed set.

For the LT model, we show the experimental results in Figure 7. Again, Greedy1 requires the most running time. However, different from the results for the IC model, Greedy2's efficiency is comparable with that of other heuristic algorithms.

3) *Memory*: In order to evaluate the space utility of our greedy algorithms, we conducted the experiments that take the memory occupied by each algorithm as a measurement.

Same as the experiments for quality and efficiency testing, we vary J and the experimental results are shown in Figure 5 for the IC model and Figure 8 for the LT model. According to these results, our greedy algorithms share the nice feature of low space complexity with other heuristic algorithms (less than 2 MB for all experiments in this paper).

4) *Error Analysis*: To verify the error bounds derived in this paper, we also conducted the experiments which compare the number of seeds returned by our algorithms with the optimal one on small datasets (0.5% of the HEP-T dataset). We performed Brute-Force searching to obtain the optimal solution.

For the IC model, the experimental results are shown in Figure 9. According to these results, the additive errors incurred by our algorithms are generally much smaller than the theoretical error bounds on the real dataset. In Figure 9(b), we find that the multiplicative error of our greedy algorithm grows slowly when J increases. Besides, we discover that k_2 is the smallest among k_1 , k_2 and k_3 in most cases of our experiments. That is, the multiplicative bound becomes $(1 + k_2)$ (i.e., $(1 + \ln \frac{\sigma'(S_1)}{\sigma'(S_h) - \sigma'(S_{h-1})})$) in these cases. Based on this, we can explain the phenomenon in Figure 9(b) that the theoretical multiplicative error bound does not change too much when we increase J from 0.75 to 1.

For the LT model, the results are shown in Figure 10. According to these results, the additive errors of our greedy algorithms are much smaller than the theoretical error bounds.

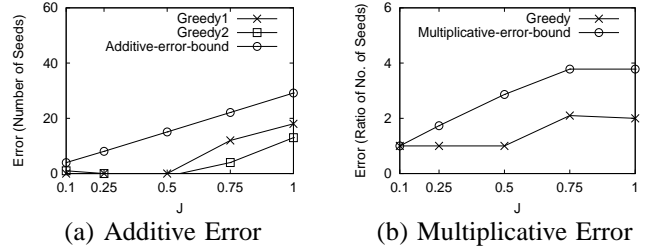


Fig. 9. Error Analysis (IC Model)

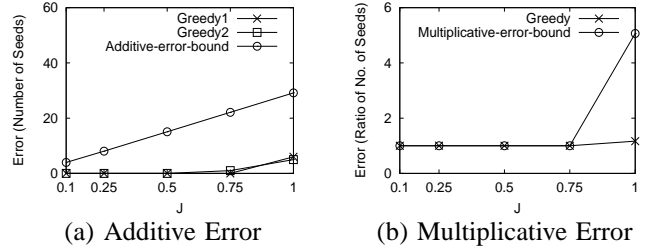


Fig. 10. Error Analysis (LT Model)

For the multiplicative errors shown in Figure 10(b), we find that the theoretical bounds are usually 1, i.e., the approximate solution should be exactly the optimal one, which is verified by our results.

5) *Full Coverage Experiments*: We conducted the experiments on our four real datasets for the Full-Coverage problem as follows. We denote by *FC- alg* our algorithm for Full-Coverage, which is described in Section VI-B. In the experiments, we compared FC- alg with Greedy1 and Greedy2 (J is set to $|V|$) in terms of the number of seeds returned the algorithm, the running time and the occupied memory.

The experimental results are shown in Figure 11. According to the results in Figure 11(a), we find that FC- alg returns less seeds than the greedy algorithms (i.e., Greedy1 and Greedy2). Besides, we find that FC- alg runs faster than the greedy algorithms, which is shown in Figure 11(b). In Figure 11(c), we notice that FC- alg is as space-efficient as the greedy algorithms.

Conclusion: Greedy1 and Greedy2 both give the smallest seed set compared with other algorithms Degree-Heuristic, Centrality-Heuristic and Random. In addition, the difference between the size of a seed set returned by Greedy1 or Greedy2 and the minimum (optimal) seed size is significantly smaller than the theoretical bound. Besides, Greedy2 performs faster than Greedy1.

VIII. CONCLUSION

In this paper, we propose a new viral marketing problem called J -MIN-Seed, which has extensive applications in real world. We then prove that J -MIN-Seed is NP-hard under two popular diffusion models (i.e., the IC model and the LT model). To solve J -MIN-Seed effectively, we develop a greedy algorithm, which can provide approximation guarantees. Besides, for the special setting where J is equal to the number of all users in the social network (i.e., Full-Coverage), we design

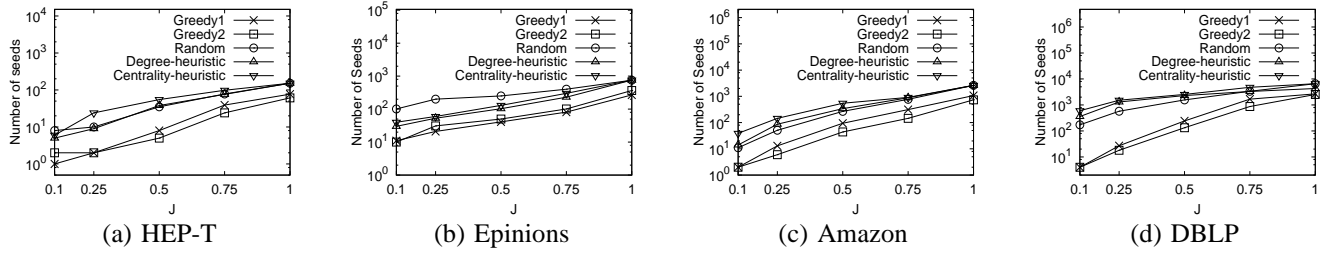


Fig. 3. Number of Seeds (IC Model)

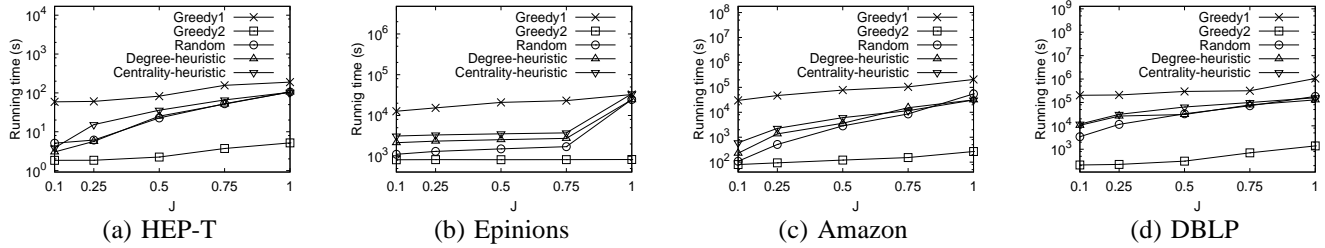


Fig. 4. Running Time (IC Model)

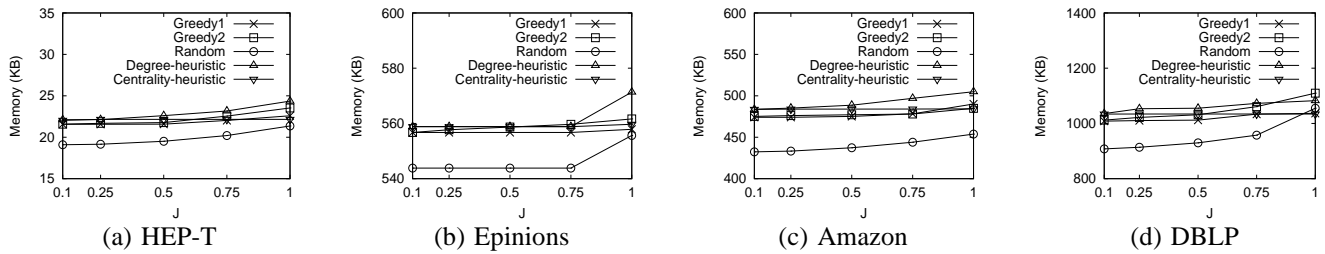


Fig. 5. Memory (IC Model)

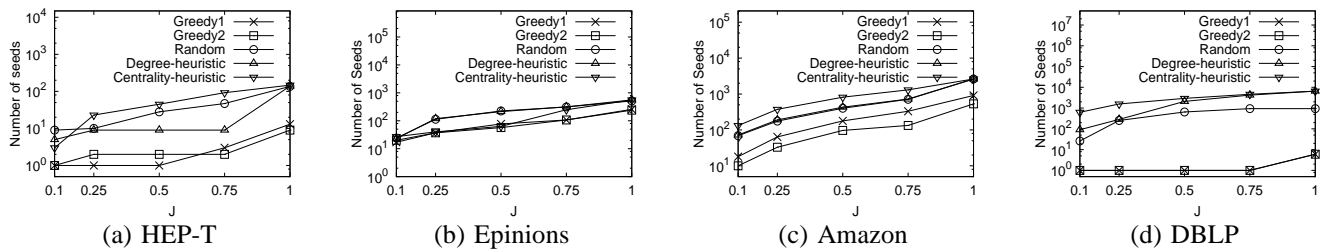


Fig. 6. Number of Seeds (LT Model)

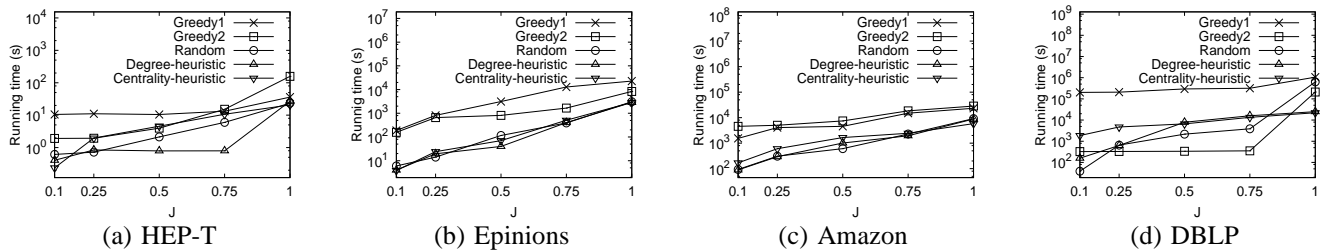


Fig. 7. Running Time (LT Model)

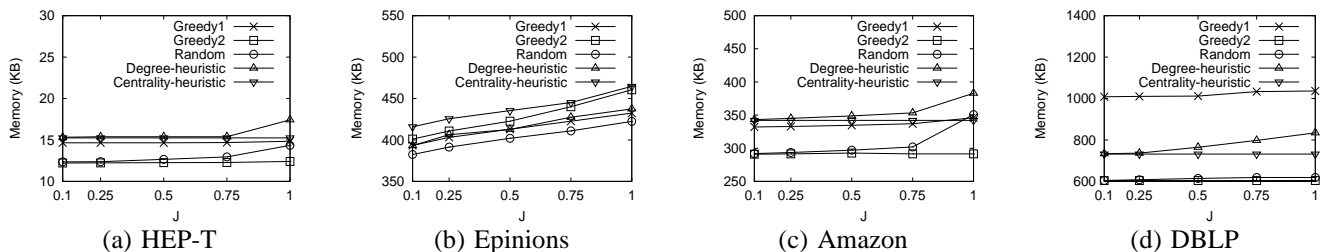


Fig. 8. Memory (LT Model)

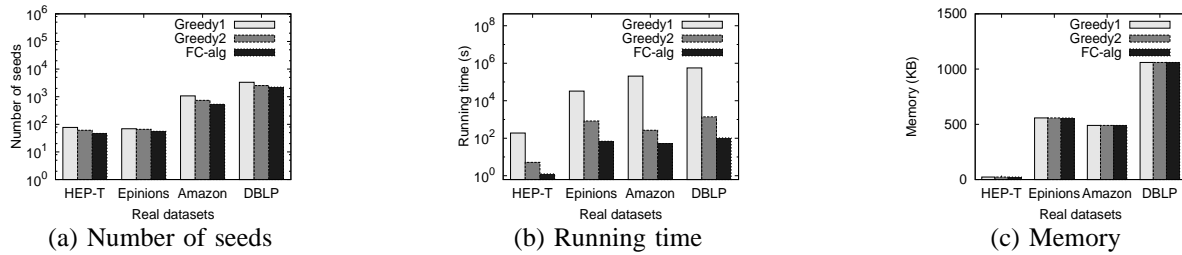


Fig. 11. Experiments for Full Coverage (Real datasets)

other efficient algorithms. Finally, we conducted extensive experiments on real datasets, which verified the effectiveness and efficiency of our greedy algorithm. For future work, we plan to study the properties of our new problem under diffusion models other than the IC model and the LT model. Finding other solutions of Full-Coverage for the LT model is another interesting direction.

Acknowledgements: The research is supported by HKRGC GRF 621309 and Direct Allocation Grant DAG11EG05G.

REFERENCES

- [1] J. Bryant and D. Miron, "Theory and research in mass communication," *Journal of communication*, vol. 54, no. 4, pp. 662–704, 2004.
- [2] J. Nail, "The consumer advertising backlash," *Forrester Research*, 2004.
- [3] I. R. Misner, *The World's best known marketing secret: Building your business with word-of-mouth marketing*. Bard Press, 2nd edition, 1999.
- [4] A. Johnson, "nike-tops-list-of-most-viral-brands-on-facebook-twitter," 2010. [Online]. Available: <http://www.kikabink.com/news/>
- [5] M. Granovetter, "Threshold models of collective behavior," *The American Journal of Sociology*, vol. 83, no. 6, pp. 1420–1443, 1978.
- [6] T. C. Schelling, *Micromotives and macrobehavior*. WW Norton and Company, 2006.
- [7] J. Goldenberg, B. Libai, and E. Muller, "Talk of the network: A complex systems look at the underlying process of word-of-mouth," *Marketing Letters*, vol. 12, no. 3, pp. 211–223, 2001.
- [8] —, "Using complex systems analysis to advance marketing theory development: Modeling heterogeneity effects on new product growth through stochastic cellular automata," *Academy of Marketing Science Review*, vol. 9, no. 3, pp. 1–18, 2001.
- [9] D. Gruhl, R. Guha, D. Liben-Nowell, and A. Tomkins, "Information diffusion through blogspace," in *WWW*, 2004.
- [10] H. Ma, H. Yang, M. R. Lyu, and I. King, "Mining social networks using heat diffusion processes for marketing candidates selection," in *CIKM*, 2008.
- [11] P. Domingos and M. Richardson, "Mining the network value of customers," in *KDD*, 2001.
- [12] D. Kempe, J. Kleinberg, and E. Tardos, "Maximizing the spread of influence through a social network," in *SIGKDD*, 2003.
- [13] J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. VanBriesen, and N. Glance, "Cost-effective outbreak detection in networks," in *SIGKDD*, 2007.
- [14] M. Kimura and K. Saito, "Tractable models for information diffusion in social networks," *PKDD*, 2006.
- [15] W. Chen, C. Wang, and Y. Wang, "Scalable influence maximization for prevalent viral marketing in large-scale social networks," in *SIGKDD*, 2010.
- [16] S. Datta, A. Majumder, and N. Shrivastava, "Viral marketing for multiple products," in *ICDM*, 2010.
- [17] M. Richardson and P. Domingos, "Mining knowledge-sharing sites for viral marketing," in *SIGKDD*, 2002.
- [18] S. Bharathi, D. Kempe, and M. Salek, "Competitive influence maximization in social networks," *Internet and Network Economics*, pp. 306–311, 2007.
- [19] T. Carnes, C. Nagarajan, S. M. Wild, and A. van Zuylen, "Maximizing influence in a competitive social network: a follower's perspective," in *Proceedings of the ninth international conference on Electronic commerce*. ACM, 2007, pp. 351–360.
- [20] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to algorithms*. The MIT press, 2009.
- [21] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher, "An analysis of approximations for maximizing submodular set functions-I," *Mathematical Programming*, vol. 14, no. 1, pp. 265–294, 1978.
- [22] L. A. Wolsey, "An analysis of the greedy algorithm for the submodular set covering problem," *COMBINATORIA*, vol. 2, no. 4, pp. 385–393, 1981.

APPENDIX: PROOF OF LEMMAS/THEOREMS

Proof of Theorem 1. We first give the J -MIN-Seed's decision problem as follows. Given a social network $G(V, E)$, an integer J and an integer l , we want to find a set S of seeds such that $|S| \leq l$ and $\sigma(S) \geq J$.

Next, we prove that this decision problem is NP-hard for the IC model.

Consider an instance of the NP-complete problem, *Maximum Cover*. Given a ground set $U = \{u_1, u_2, \dots, u_n\}$ and a collection of its subsets $C = \{S_1, S_2, \dots, S_m\}$. Let k and B be two integers. We want to know whether there exists a subset $C' \subseteq C$ such that $|C'| \leq k$ and $|\cup_{S \in C'} S| \geq B$.

We transform the aforementioned Maximum Cover problem to our J -MIN-Seed's decision problem as follows. We first construct a set V of nodes and a set E of edges as follows. V is constructed as follows. Initially, V is set to an empty set. For each element u_j in U , we create a node v_{u_j} and insert it into V . Besides, for each subset S_i in C , we create a node v_{S_i} and insert it into V . Then, we construct E as follows. Initially, E is set to an empty set. For each set S_i in C and each element u_j in S_i , we construct an edge (v_{S_i}, v_{u_j}) and insert it into E . The weight of each edge in E is set to 1. At the end, we obtain a graph G with a set V of nodes and a set E of edges. We set l to be k and J to be $B + k$. Clearly, the above transformation step runs in polynomial time.

We show that the Maximum Cover problem is equivalent to deciding whether there is a seed set S such that $|S| \leq k$ and $\sigma(S) \geq B + k$. If the Maximum Cover problem is solvable (i.e., there exists a subset C' of C such that $|C'| \leq k$ and at least B elements in U are covered by C'), then the seed set containing all the nodes corresponding to the subsets in C' incurs at least $B + k$ influenced nodes. Hence, the J -MIN-Seed's decision problem is also solvable. Similarly, we can see that the reverse process also works. Thus, J -MIN-Seed is NP-hard.

Now, we prove the analogous result for the LT model. We construct a special case of the J -MIN-Seed's decision problem for the LT model by specifying the thresholds of all nodes

to be 0s. Consequently, the influence propagation process is identical to the aforementioned special instance defined for the IC model (where the weight of each edge is 1). So, we can draw the same conclusion for the LT model. ■

Proof of Lemma 1. Assume that we perform the social network sampling process (on G) n times and thus obtain n sampled graphs of G . We denote these sampled graphs by G_1, G_2, \dots, G_n . Let I_i be the influence incurred by the seed set S on the sampled graph G_i . Let $E(I)$ be the expected value of I_i and $\bar{I} = \sum_{i=1}^n I_i$ be the mean of the I_i values on the sampled graphs. According to *Hoeffding's Inequality*, for any non-negative real number t , we know

$$Pr(|\bar{I} - E(I)| \geq t) \leq 2 \exp\left(-\frac{2t^2n^2}{\sum_{i=1}^n (u_i - l_i)^2}\right)$$

where u_i and l_i are the upper bound and the lower bound of I_i , respectively.

Considering $I_i \leq |V|$ and $I_i \geq 1$, i.e., $u_i = |V|$ and $l_i = 1$, for $1 \leq i \leq n$, we have

$$Pr\left(\left|1 - \frac{\bar{I}}{E(I)}\right| \geq \frac{t}{E(I)}\right) \leq 2 \exp\left(-\frac{2t^2n}{(|V| - 1)^2}\right)$$

Let $\epsilon = t/E(I)$. We obtain

$$Pr\left(\left|1 - \frac{\bar{I}}{E(I)}\right| \geq \epsilon\right) \leq 2 \exp\left(-\frac{2\epsilon^2 E(I)^2 n}{(|V| - 1)^2}\right)$$

Hence, in order to obtain a $(1 \pm \epsilon)$ -approximation algorithm with the confidence at least c , the following condition should hold.

$$2 \exp\left(-\frac{2\epsilon^2 E(I)^2 n}{(|V| - 1)^2}\right) \leq 1 - c$$

As a result, we obtain the requirement on the number of times for sampling as follows.

$$n \geq \frac{(|V| - 1)^2 \ln\left(\frac{2}{1-c}\right)}{2\epsilon^2 E(I)^2}$$

Since the seeds themselves would be influenced, we know that $E(I) \geq |S|$. As a result, we obtain the following inequality.

$$n \geq \frac{(|V| - 1)^2 \ln\left(\frac{2}{1-c}\right)}{2\epsilon^2 |S|^2}$$

Thus, we finish our proof. ■

Proof of Lemma 2. Firstly, we give the theoretical bound on the influence for k -MAX-Influence. The problem of determining the k -element set $S \subset V$ that maximizes the value of $\sigma(\cdot)$ is NP-hard. Fortunately, according to [21], a simple greedy algorithm can solve this maximization problem with the approximation factor of $(1 - 1/e)$ by initializing an empty set S and iteratively adding the node such that the marginal gain of inserting this node into the current set S is the greatest one until k nodes have been added. We present this interesting tractability property of maximizing a submodular function in Lemma 4 as follows.

Lemma 4 ([21]): For a non-negative, monotone submodular function f , we obtain a set S of size k by initializing set

S to be an empty set and then iteratively adding the node u one at a time such that the marginal gain of inserting u into the current set S is the greatest. Assume that S^* is the set with k elements that maximizes function f , i.e., the optimal k -element set. Then, $f(S) \geq (1 - 1/e) \cdot f(S^*)$, where e is the natural logarithmic base. ■

Secondly, we derive the additive error bound on the seed set size for J -MIN-Seed based on the aforementioned bound.

As discussed in Section III, $\sigma(\cdot)$ is submodular. Clearly, $\sigma(\cdot)$ is also non-negative and monotone. The framework in Algorithm 1 involves a number of iterations (lines 2-4) where the size of the seed set S is incremented by one for each iteration. We say that the framework in Algorithm 1 is at stage j if the seed set S contains j seeds at the end of an iteration. The seed set S at stage j is denoted by S_j . Consequently, according to Lemma 4, at each stage j , we conclude that

$$\sigma(S_j) \geq (1 - 1/e) \cdot \sigma(S_j^*) \quad (1)$$

where S_j^* is the set that provides the maximum value of $\sigma(\cdot)$ over all possible seed sets of size j .

Note that the total number of stages for the greedy process is equal to h (i.e., the size of the seed set returned by the algorithm). That is, the greedy process stops at stage h . Thus, we know that $\sigma(S_h) \geq J$ and the greedy solution for J -MIN-Seed is S_h . Consider the last two stages, namely stage $h - 1$ and stage h . We know that $\sigma(S_{h-1}) < J$ and $\sigma(S_h) \geq J$. Since $\sigma(S_h^*) \geq \sigma(S_h)$, we have $\sigma(S_h^*) \geq J$.

Now, we want to explore the relationship between h and t . Note that the following inequality holds.

$$t \leq h \quad (2)$$

Consider two stages, stage i and stage $i + 1$, such that $\sigma(S_i) < (1 - 1/e) \cdot J$ while $\sigma(S_{i+1}) \geq (1 - 1/e) \cdot J$. According to Inequality (1), we know $\sigma(S_i^*) < J$. (This is because if $\sigma(S_i^*) \geq J$, then we have $\sigma(S_i) \geq (1 - 1/e) \cdot J$ with Inequality (1), which contradicts $\sigma(S_i) < (1 - 1/e) \cdot J$.) As a result, we have the following inequality

$$t > i \quad (3)$$

due to the monotonicity property of $\sigma(\cdot)$.

According to Inequality (2) and Inequality (3), we obtain $t \in [i + 1, h]$. That is, the additive error of our greedy algorithm (i.e., $h - t$) is bounded by the number of stages between stage $i + 1$ and stage h . Since $\sigma(S_{i+1}) \geq (1 - 1/e) \cdot J$ and $\sigma(S_{h-1}) < J$, the difference of the influence incurred between stage $i + 1$ and stage $h - 1$ is bounded by $J - (1 - 1/e) \cdot J = 1/e \cdot J$. Since each stage increases at least 1 influenced node (seed itself), it is easy to see that the number of stages between stage $i + 1$ and stage $h - 1$ is at most $1/e \cdot J$. Consequently, the number of stages between stage $i + 1$ and stage h is at most $1/e \cdot J + 1$. As a result, $h - t \leq 1/e \cdot J + 1$. ■

Proof of Lemma 3. This proof involves four parts. In the first part, we construct a new problem P' based on the submodular function $\sigma'(\cdot)$ (instead of $\sigma(\cdot)$). In the second part, we show the multiplicative error bound of the greedy

algorithm in Algorithm 1 (using $\sigma'(\cdot)$ instead of $\sigma(\cdot)$) for this new problem P' . We denote this adapted greedy algorithm by A' . For simplicity, we denote the original greedy algorithm in Algorithm 1 using $\sigma(\cdot)$ by A . In the third part, we show that this new problem is equivalent to the J -MIN-Seed problem. In the fourth part, we show that the multiplicative error bound deduced in the second part can be used as the multiplicative error bound of algorithm A for J -MIN-Seed.

Firstly, we construct a new problem P' as follows. Note that $\sigma'(S) = \min\{\sigma(S), J\}$. Problem P' is formalized as follows.

$$\arg \min\{|S| : \sigma'(S) = \sigma'(V), S \subseteq V\}. \quad (4)$$

Secondly, we show the multiplicative error bound of algorithm A' for problem P' by using the following Lemma 5 [22].

Lemma 5 ([22]): Given problem $\arg \min\{\sum_{x \in S} g(x) : f(S) = f(U), S \subseteq U\}$ where f is a nondecreasing and submodular function defined on subsets of a finite set U , and g is a function defined on U . Consider the greedy algorithm that selects x in $U - S$ such that $(f(S \cup \{x\}) - f(S))/g(x)$ is the greatest and adds it into S at each iteration. The process stops when $f(S) = f(U)$. Assume that the greedy algorithm terminates after h iterations and let S_i denote the seed set at iteration i ($S_0 = \emptyset$). The greedy algorithm provides a $(1 + \min\{k_1, k_2, k_3\})$ -approximation of the above problem, where $k_1 = \ln \frac{f(U) - f(\emptyset)}{f(U) - f(S_{h-1})}$, $k_2 = \ln \frac{f(S_1) - f(\emptyset)}{f(S_h) - f(S_{h-1})}$, and $k_3 = \ln(\max\{\frac{f(\{x\}) - f(\emptyset)}{f(S_i \cup \{x\}) - f(S_i)} | x \in U, 0 \leq i \leq h, f(S_i \cup \{x\}) - f(S_i) > 0\})$. ■

We apply the above lemma for problem P' as follows. It is easy to verify that $\sigma'(\cdot)$ is a non-decreasing and submodular function defined on subsets of a finite set V . We set U to be V and set $f(\cdot)$ to be $\sigma'(\cdot)$. We also define $g(x)$ to be 1 for each $x \in V$ (or U). Note that $\sum_{x \in S} g(x) = |S|$. We re-write Problem P' (4) as follows.

$$\arg \min\{\sum_{x \in S} g(x) : \sigma'(S) = \sigma'(V), S \subseteq V\}. \quad (5)$$

The above form of problem P' is exactly the form of the problem described in Lemma 5. Suppose that we adopt the greedy algorithm in Algorithm 1 for problem P' by using $\sigma'(\cdot)$ instead of $\sigma(\cdot)$, i.e., algorithm A' . It is easy to verify that algorithm A' follows the steps of the greedy algorithm described in Lemma 5 (i.e., selecting the node x such that $(\sigma'(S \cup \{x\}) - \sigma'(S))/g(x)$ is the greatest where $g(x)$ is exactly equal to 1). By Lemma 5, the greedy algorithm A' for problem P' gives $(1 + \min\{k_1, k_2, k_3\})$ -approximation of problem P' , where $k_1 = \ln \frac{\sigma'(V) - \sigma'(\emptyset)}{\sigma'(V) - \sigma'(S_{h-1})} = \ln \frac{J}{J - \sigma'(S_{h-1})}$, $k_2 = \ln \frac{\sigma'(S_1) - \sigma'(\emptyset)}{\sigma'(S_h) - \sigma'(S_{h-1})} = \ln \frac{\sigma'(S_1)}{\sigma'(S_h) - \sigma'(S_{h-1})}$, and $k_3 = \ln(\max\{\frac{\sigma'(\{x\})}{\sigma'(S_i \cup \{x\}) - \sigma'(S_i)} | x \in V, 0 \leq i \leq h, \sigma'(S_i \cup \{x\}) - \sigma'(S_i) > 0\})$.

Thirdly, we show that problem P' is equivalent to the J -MIN-Seed problem which can be formalized as follows (since $\sum_{x \in S} g(x) = |S|$).

$$\arg \min\{\sum_{x \in S} g(x) : \sigma(S) \geq J, S \subseteq V\}. \quad (6)$$

In the following, we show that the set of all possible solutions for the problem in form of (6) (i.e., the J -MIN-Seed problem) is equivalent to the set of all possible solutions for the problem in form of (5) (i.e., problem P'). Note that the objective functions in both problems are equal. The remaining issue is to show that the constraints for one problem are the same as those for the other problem.

Suppose that S is a solution for the problem in form of (6). We know that $\sigma(S) \geq J$ and $S \subseteq V$. We derive that $\sigma'(S) = J$. Since $\sigma'(V) = J$, we have $\sigma'(S) = \sigma'(V)$ and $S \subseteq V$ (which are the constraints for the problem in form of (5)).

Suppose that S is a solution for the problem in form of (5). We know that $\sigma'(S) = \sigma'(V)$ and $S \subseteq V$. Since $\sigma'(V) = J$, we have $\sigma'(S) = J$. Considering $\sigma'(S) = \min\{\sigma(S), J\}$, we derive that $\sigma(S) \geq J$. So, we have $\sigma(S) \geq J$ and $S \subseteq V$ (which are the constraints for the problem in form of (6)).

Fourthly, we show that the size of the solution (i.e., $|S|$) returned by algorithm A' for the new problem P' is equal to that returned by algorithm A for J -MIN-Seed. Since $\sigma(S_i) < J$ for $1 \leq i \leq h-1$, we know that $\sigma'(S_i) = \sigma(S_i)$ for $1 \leq i \leq h-1$. We also know that the element x in $V - S_{i-1}$ that maximizes $\sigma(S_{i-1} \cup \{x\}) - \sigma(S_{i-1})$ (which is chosen at iteration i by algorithm A) would also be the element that maximizes $\sigma'(S_{i-1} \cup \{x\}) - \sigma'(S_{i-1})$ (which is chosen at iteration i by algorithm A') for $i = 1, 2, \dots, h-1$. That is, algorithm A' would proceed in the same way as algorithm A at iteration $i = 1, 2, \dots, h-1$. Consider iteration h of algorithm A . We denote the element selected by algorithm A by x_h . Then, we know $\sigma(S_{h-1} \cup \{x_h\}) \geq J$ since algorithm A stops at iteration h . Consider iteration h of algorithm A' . This iteration is also the last iteration of A' . This is because there exists an element x in $V - S_{h-1}$ such that $\sigma'(S_{h-1} \cup \{x\}) = \sigma'(V) (= J)$ (since x can be equal to x_h where $\sigma'(S_{h-1} \cup \{x_h\}) = J$). Note that this element x maximizes $\sigma'(S_{h-1} \cup \{x\}) - \sigma'(S_{h-1})$ and thus is selected by A' . We conclude that both algorithms A and A' terminates at iteration h . Since the number of iterations for an algorithm (A or A') corresponds to the size of the solution returned by the algorithm, we deduce that the size of the solution returned by algorithm A' is equal to that returned by algorithm A .

In view of the above discussion, we know that problem P' is equivalent to J -MIN-Seed and algorithm A' for problem P' would proceed in the same way as algorithm A for J -MIN-Seed. As a result, the multiplicative bound of algorithm A' for problem P' in the second part also applies to algorithm A (i.e., the greedy algorithm in Algorithm 1) for J -MIN-Seed. ■