

# Revenue Maximization by Viral Marketing: A Social Network Host’s Perspective

Arijit Khan<sup>1</sup>, Benjamin Zehnder<sup>2</sup>, Donald Kossmann<sup>2</sup>  
<sup>1</sup>Nanyang Technological University, Singapore    <sup>2</sup>ETH Zurich, Switzerland

**Abstract**—We study the novel problem of revenue maximization of a social network host that sells viral marketing campaigns to multiple competing campaigners. Each client campaigner informs the social network host about her target users in the network, as well as how much money she is willing to pay to the host if one of her target users buys her product. The social network host, in turn, assigns a set of seed users to each of her client campaigners. The seed set for a campaigner is a limited number of users to whom the campaigner provides free samples, discounted price etc. with the expectation that these seed users will buy her product, and would also be able to influence many of her target users in the network towards buying her product. Because of various product-adoption costs, it is very unlikely that an average user will purchase more than one of the competing products. Therefore, from the host’s perspective, it is important to assign seed users to client campaigners in such a way that the seed assignment guarantees the maximum aggregated revenue for the host considering all her client campaigners.

We formulate our problem by following two well-established influence cascading models: the independent cascade model and the linear threshold model. While our problem using both these models is NP-hard, and neither monotonic, nor sub-modular; we develop approximated algorithms with theoretical performance guarantees. However, as our approximated algorithms often incur higher running times, we also design efficient heuristic methods that empirically perform as good as our approximated algorithms. Our detailed experimental evaluation attests that the proposed techniques are effective and scalable over real-world datasets.

## I. INTRODUCTION

In viral marketing, whenever a social network user buys a product, she is viewed as being influenced or activated. The classical viral marketing problem [9], [13] identifies the top- $k$  seed users in a social network such that the expected number of influenced users in the network, starting from those seed users and following some influence cascading model, is maximized. The budget  $k$  on the seed-set size usually depends on the campaigner — in other words, it depends on how many initial users the campaigner can directly influence to buy her product by advertisements, giving free samples, and discounted prices.

The bulk of the research in the domain of viral marketing assumes that the social network structure is available to the campaigners. However, in real-world scenarios, the social network platforms are owned by third-party hosts [19], such as Facebook, Twitter, and LinkedIn; and the hosts keep their social graphs secret for their own benefits and for privacy reasons. Therefore, marketing companies themselves are not able to select their best seed sets due to lack of access to the social network graph.

In this study, we assume that the seed set selections are done by the social network host on behalf of her clients, who are the marketing campaigners. The campaigners, on the other hand, spend their overall budget for viral marketing into two

parts. Particularly, each campaigner informs the host about: (a) her budget on the seed-set size (i.e., the number of seed users,  $k$ ), and also (b) how much money she is willing to pay to the host for each of her target users if that user adopts her product. While the campaigner might not know the exact social network structure, it is usually easier for her to define her target users, either explicitly, or via some constraints, e.g., people in the age group 20-30, all banking professionals, etc. We note that the number of such target users for a campaigner can be very large, and it is often not possible (or not economical) to give each of them a free sample or discounted price. Therefore, the campaigner still allocates a small  $k$  as the number of her seed nodes. She uses rest of her budget to pay the social network host according to the agreement, which can be a small amount of money for each of her target users who adopts her product.

In real-world, multiple companies compete and they launch comparable products around the same time <sup>1</sup> (e.g., Nintendo’s Wii vs. Sony’s Playstation vs. Microsoft’s X-Box; Microsoft’s Surface vs. Apple’s iPad vs. Samsung Note 3) [18], [19]. Thus, the host often needs to run multiple competing viral marketing campaigns together over the network. However, due to various product-adoption costs, it is very unlikely that an average user will purchase more than one of the competing products. Since most of the users adopt only one of the competing products, it implies that the seed sets of the competing campaigners require to be mutually non-overlapping [18], [19]. Therefore, from the host’s perspective, the challenge lies in how to select the seed set for each of her client campaigners so that the host’s overall expected revenue is maximized.

Running multiple viral marketing campaigns by a social network host was studied earlier in [19] by Lu et. al. However, [19] studied a different problem. The problem our paper addresses is the problem of maximizing the revenue of the host of the social network. In contrast, [19] studied how to balance the expected spread of each campaign over the network, which we believe is less relevant in practice. When a social network host is selecting the seed sets on behalf of her client campaigners, maximizing the host’s overall expected revenue is the problem of interest for most practical scenarios. Furthermore, [19] makes a number of additional assumptions. For instance, [19] did not apply the notion of target users for each client campaigner; they assume that all users in the network are equally important to all campaigners. In reality, a campaigner often promotes her product with a group of target customers in mind [15], [17], and is willing to pay more money to the host if her target users adopt her product. Indeed, as shown experimentally in [1], a product adopted by customers (or, a campaign reached to users) outside the target group could

---

<sup>1</sup>At the Consumer Electronic Show in January 2011, over 80 new tablets were announced by Motorola, Samsung, and Toshiba. (<http://mashable.com/2011/01/12/ces-2011-tablet-videos/>)

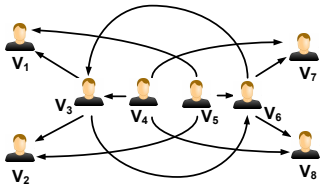


Fig. 1: Revenue Maximization: Limitations of Naïve Methods

generate negative impression towards the campaigner, which will affect her image in the long run. In summary, for different users, each campaigner would be willing to pay a different amount of money to the host if those users buy her product.

We next demonstrate with an example why the method in [19] and other naïve approaches are not suitable for our problem.

*Example 1:* Assume that there are two campaigners —  $C_1$  and  $C_2$ , who are viral marketing clients to the social network host as depicted in Figure 1. We also assume that each directed edge has an influence probability 1, and we apply the independent cascade (IC) model [13], described later in Section III-B. The model assumes that the cascade of influence happens in discrete time steps. Each node can be activated only once and by only one of the campaigns; also the node stays activated with that campaign until the end. Let  $A_{ij}$  denote the money that campaigner  $C_i$  is willing to pay to the host if node  $V_j$  adopts her product. We set  $A_{11} = A_{12} = A_{13} = A_{14} = \text{US\$ } 10$ ,  $A_{15} = A_{16} = A_{17} = A_{18} = \text{US\$ } 1$ ; while  $A_{21} = A_{22} = A_{23} = A_{24} = \text{US\$ } 1$ , and  $A_{25} = A_{26} = A_{27} = A_{28} = \text{US\$ } 10$ . Assume that the budget on the seed set size for each campaigner is 1.

**What is the optimal solution?** The optimal solution is as follows. If  $V_3$  and  $V_6$  are selected as the seed nodes for  $C_1$  and  $C_2$ , respectively, then  $V_1, V_2, V_3$  will be influenced by  $C_1$ , while  $V_6, V_7$ , and  $V_8$  will be influenced by  $C_2$ . This will ensure an aggregated revenue of US\$ 60 to the host.

**Why a naïve method will not work?** A naïve approach to solve the host’s revenue maximization problem would be to first define some order among the campaigners, and then identify the top- $k$  seed nodes for each campaigner such that the host’s revenue is maximized considering one campaigner at a time. If some of the top- $k$  seed nodes for the current campaigner have already been assigned to previous campaigners, we identify the next-best seed nodes for the current campaigner until we exhaust the budget  $k$  of the current campaigner’s seed-set size. If we apply this naïve approach, we get  $V_4$  as the best seed node for  $C_1$ . This is because  $V_4$  could eventually influence  $V_1, V_2, V_3, V_6, V_7$ , and  $V_8$ , and the host will get a revenue of US\$ 43, assuming  $C_1$  is the only campaigner in the network. Similarly, we find that  $V_5$  is the best seed node for  $C_2$ , assuming there is no other campaigner. Now, in reality, when the host runs the two viral marketing campaigns simultaneously with campaign of  $C_1$  starting from  $V_4$  and that of  $C_2$  starting from  $V_5$ , the host’s aggregate revenue will be only US\$ 44. This is because after simultaneous campaigning,  $V_3, V_4, V_7$ , and  $V_8$  will be influenced by  $C_1$ , while the remaining nodes will be influenced by  $C_2$ .

**Why the method in [19] will not work?** Lu et. al.’s problem formulation [19] identifies the seed sets for the campaigners in a way such that the expected spread of each campaign is almost

equal (i.e., maintaining fairness), while also maximizing the overall spread of all campaigns in the network. As an example, selection of  $V_4$  as the seed node of  $C_1$  and  $V_5$  as the seed node of  $C_2$  would be an optimal solution according to [19], since this will result in  $V_3, V_4, V_7$ , and  $V_8$  to be influenced by  $C_1$ , while the remaining four nodes will be influenced by  $C_2$ . Note that the host’s revenue in this solution is only US\$ 44.

The above example clearly illustrates that the aforementioned naïve approach as well as [19] may result in a sub-optimal aggregated revenue for the host. In fact, as [19] did not consider the notion of target users for each client campaigner, it is non-trivial to extend their proposed algorithm to solve our current problem. Specifically, their **Needy-Greedy** algorithm heuristically partitions the initial seed set to balance the expected influence spread of each campaigner, assuming that all users in the network are equally important to all campaigners. It is difficult to adapt such an algorithm to maximize the host’s overall revenue from all campaigners in a scenario when each user has a different importance to every campaigner. Indeed, our dynamic programming based seed partitioning strategy introduced in Section 5.2 is different from their Needy-Greedy heuristics.

**Our Contributions and Roadmap.** Our contributions can be summarized as follows:

- We define the fundamental problem of host’s revenue maximization by viral marketing in the presence of  $m \geq 2$  competitive campaigners (Sec. III).
- We formulate the problem using two widely-used influence cascading models — the independent cascade (IC) model (Section IV) and the linear threshold (LT) model (Sec. V).
- We show that our problem using both these models is NP-hard, and neither monotonic, nor sub-modular. We therefore develop approximated algorithms to solve our problem. In addition, we also design more efficient and scalable heuristic techniques that empirically perform as good as our approximated algorithms.
- We conduct a thorough experimental evaluation using several real-world datasets and with various kinds of revenue distributions (Sec. VII). Our empirical results attest that the proposed methods efficiently generate high-quality results.

## II. RELATED WORK

In viral marketing, a social network user is considered influenced or activated by a campaign if she buys a product corresponding to the campaign. The classical viral marketing problem aims at finding a small number of seed nodes that generates the largest expected influence cascade in a social network. Domingos and Richardson [9] formulated viral marketing as an optimization problem. Kempe et. al. [13] proposed the linear threshold model and the independent cascade model, and designed approximation algorithms with provable performance guarantees. However, the computation of influence cascade is still #P-hard [7]. Several heuristics have been proposed to improve the efficiency of viral marketing [8], [11]. Very recently, [22] developed almost linear-time viral marketing algorithms, yet providing the same approximation guarantee as Kempe et. al.’s original method. In [15], Lappas

et. al. introduced the concept of target marketing and  $k$ -effectors — by identifying  $k$  seed nodes such that a given activation pattern can be established. The notion of target marketing was also considered in [17] that maximizes influence over a region of the network.

Viral marketing in the presence of a negative campaign was investigated in [2], [5]. These works assume that the later campaign has prior knowledge of rival side’s initial seed nodes. Bordin et. al. [3] analyzed the similar problem under the LT model; while [4], [6] attempt at preventing the spread of an existing negative campaign in the network. Recently, [23] studied the viral marketing problem between non-cooperative campaigns who select seeds alternatively. However, as discussed in Section 1, competitive new products from rival companies are often launched around the same time. Thus, [12], [18], [19] considered viral marketing in the presence of multiple competing campaigners, who promote their products in a social network around the same time. We also consider a similar scenario, i.e., multiple rival companies launch and promote competing products at the same time.

While the bulk of the research on viral marketing assumes that the social network structure is available to the campaigners; in reality, the social network platforms are owned by third-party hosts. Lu et al. [19] were the first to consider the viral marketing problem from the social network host’s perspective. In their framework, the host selects the seed nodes on behalf of her client campaigners so that the expected influence spread for each client campaigner becomes nearly the same. However, when a social network host is selecting the seed sets on behalf of her client campaigners, maximizing the host’s overall expected revenue would be the problem of interest for most practical scenarios. To the best of our knowledge, ours is the first work that studies the host’s revenue maximization problem, while also considering the notion of target users for each campaigner. As [19] did not consider the notion of target users for each client campaigner, it is non-trivial to extend their proposed algorithm to solve our current problem.

### III. PRELIMINARIES

#### A. Problem Statement

A social network  $\mathcal{G}$  is modeled as a triple  $(V, E, P)$ , where  $V$  is a set of  $n$  nodes,  $E \subseteq V \times V$  is a set of  $e$  directed edges, and  $P : E \rightarrow (0, 1)$  is a probability function that assigns a probability to each edge in  $E$ . The probability  $p_{uv}$  on a directed edge  $(u, v) \in E$  represents the probability that node  $v$  adopts a product due to the influence of node  $u$ , because  $u$  adopted that product before. When  $v$  adopts that product, it automatically becomes eligible to influence its neighbors who has not adopted that product already. We shall discuss the details of various influence cascading models in Section III-B.

We consider  $m \geq 2$  competing campaigners, denoted by  $C_1, C_2, \dots, C_m$ , for whom the social network host runs simultaneous viral marketing campaigns. We denote by  $S_i$  the seed set for campaigner  $C_i$ , and  $A_{iu}$  the money that campaigner  $C_i$  is willing to pay to the host if node  $u$  adopts  $C_i$ ’s product. We refer to  $A = (A_{iu})_{m \times n}$  the *revenue matrix*. We denote by  $Pr(u, i, S)$  the probability that node  $u$  will adopt  $C_i$ ’s product due to the influence of her campaign by following some influence cascading model, where  $S = \{S_1, S_2, \dots, S_m\}$  represents the seed sets for the  $m$  campaigners. We are now ready to define our problem.

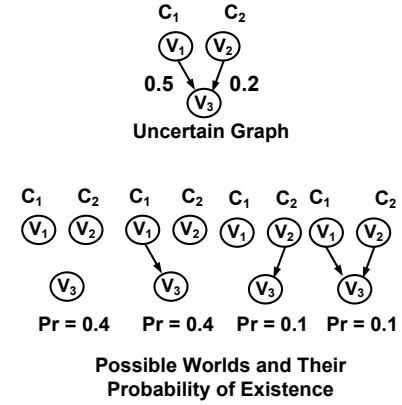


Fig. 2: Example of MCIC model

**Problem 1 (Revenue Maximization):** Given a network  $\mathcal{G} = (V, E, P)$ ,  $m \geq 2$  client campaigners, the revenue matrix  $A$ , and a budget  $k_i$  on the seed set size for each campaigner  $C_i$ , i.e.,  $|S_i| = k_i$  for  $1 \leq i \leq m$ , find the seed set for each campaigner such that the expected revenue of the host is maximized. Formally,

$$\arg \max_{S_1, S_2, \dots, S_m} \sum_{i=1}^m \sum_{u \in V} [A_{iu} \cdot Pr(u, i, S)]$$

$$\text{such that } |S_i| = k_i \quad \forall i \in (1, m)$$

$$\text{and } S_i \cap S_j = \phi \quad \forall i \neq j; \quad i, j \in (1, m) \quad (1)$$

#### B. Influence Cascading Models

We apply two widely-used influence cascading models: independent cascade (IC) and linear threshold (LT) [13].

**1) Independent Cascade Model:** In the single-campaigner IC model, the campaign starts with an initially active (i.e., adopted her product) set of seed nodes, and then unfolds in discrete steps. When some node  $u$  first becomes active at step  $t$ , it gets a single chance to activate each of its currently inactive out-neighbors  $v$ ; it succeeds with probability  $p_{u,v}$ . If  $u$  succeeds, then  $v$  will become active at step  $t+1$ . Whether or not  $u$  succeeds at step  $t$ , it cannot make any further attempts in the subsequent rounds. If a node  $v$  has incoming edges from multiple newly activated nodes, their attempts are sequenced in an arbitrary order. Also, each node can be activated only once and it stays active until the end. The campaigning process runs until no more activations are possible.

**Multi-Campaigner Independent Cascade Model.** We shall now introduce the multi-campaigner Independent Cascade (MCIC) model [4], which models multiple campaigns that are being run simultaneously in a network. MCIC follows the same process as IC, except two major differences. First, if some node  $u$  is activated with campaign of  $C_i$ , it attempts to activate its out-neighbors  $v$  with the campaign of  $C_i$ . Second, an activated node  $v$  adopts one campaign uniform at random from all its in-neighbors which were successfully activated in the last round. Each node can be activated only once and by only one of the campaigns; also the node stays activated with that campaign until the end. Therefore, the MCIC model assumes the following influence cascading scenario: people adopt a product when they come in direct contact with their friends who very recently adopted that product.

*Example 2 (MCIC Model):* In Figure 2, we show a social network along with edge probabilities. We also assume that  $V_1$  and  $V_2$  are seed nodes for campaigners  $C_1$  and  $C_2$ , respectively, and we want to calculate the probability that node  $V_3$  will be influenced by each of these campaigners using the MCIC model. The computation is carried out following the *possible world* semantics [13]. We first identify all possible worlds of the uncertain input graph, where each possible world is a certain instance of the uncertain graph, and obtained by independent sampling of the edges. Each possible world is associated with a probability of existence. For example, the second possible world in Figure 2 has probability of existence 0.4, which is due to the presence of the edge  $V_1V_3$  with probability 0.5 and the absence of the edge  $V_2V_3$  with probability  $(1-0.2)$ . Hence, the probability of existence of the second possible world is  $0.5 \times (1-0.2) = 0.4$ . In our example, there can be total 4 possible worlds. In each possible world, a node is activated by its closest seed nodes. Thus,  $V_3$  is activated by  $C_1$  in the second possible world, by  $C_2$  in the third possible world, and by either of  $C_1$  and  $C_2$  with equal probability in the fourth possible world. Therefore, the probability that  $V_3$  is activated by  $C_1$  is  $0.4 + 0.1 \times \frac{1}{2} = 0.45$ , and  $V_3$  is activated by  $C_2$  with probability  $0.1 + 0.1 \times \frac{1}{2} = 0.15$ .

2) *Linear Threshold Model:* In the single-campaigner LT model, each node  $v$  has an activation threshold  $\theta_v \leq 1$ . In addition, there is a constraint that the sum of the probabilities of all incoming edges for every node must be at most 1. The campaign starts with an initially active set of seed nodes, and then unfolds in discrete steps. If the sum of the probabilities of the incoming edges from all active nodes is greater than or equal to the activation threshold of an inactive node, then the node gets activated in the next round. Each node can only be activated once and stays active until the end.

**Multi-Campaigner Linear Threshold Model.** The multi-campaigner LT model, also termed as the K-LT model in [19], follows two steps in each round. The first step decides whether a new node will get activated and it works in exactly the same way as the LT model (i.e., without distinguishing among multiple campaigns). However, in the second step, it decides which campaign each of those newly activated nodes will adopt. Let us consider all nodes  $u$  that were activated in the last round and contributed to the activation of a node  $v$  in the current round. Then,  $v$  will adopt the same campaign as that of  $u$  with probability  $\frac{p_{uv}}{\sum_u p_{uv}}$ . With the K-LT model, each node can be activated only once and by only one of the campaigns; also the node stays activated with that campaign until the end. One may note that the K-LT model simulates the following scenario: a user adopts a technology only when more than a threshold number of her neighbors adopted a similar technology. However, once the user decides to adopt the technology, she decides on the specific product only based on her neighbors who most recently adopted that technology.

*Example 3 (K-LT Model):* We show an example of the K-LT model in Figure 3. Here,  $V_1$  and  $V_4$  are seeds for campaigners  $C_1$  and  $C_2$ , respectively. At time step 1,  $V_2$  becomes active with campaign of  $C_1$ , since  $p_{v_1,v_2} = 0.8 > \theta_{v_2} = 0.6$ . However,  $V_3$  remains inactive as  $p_{v_4,v_3} = 0.5 < \theta_{v_3} = 0.6$ . At time step 2,  $V_3$  first gets activated as the total incoming influence from its activated neighbors is:  $p_{v_2,v_3} + p_{v_4,v_3} = 0.7$ , which is higher than its activation threshold  $\theta_{v_3} = 0.6$ . Finally,

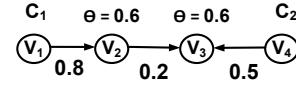


Fig. 3: Example of K-LT model

$V_3$  selects campaign of  $C_1$  with probability 1, because  $V_2$  is the only neighbor of  $V_3$  which was activated in the last round, and  $V_2$  was activated with campaign of  $C_1$ .

### C. Hardness Results

We first prove that Problem 1 is NP-hard both with the MCIC and K-LT influence cascading models. We consider the decision version of our problem: Given a social network  $\mathcal{G} = (V, E, P)$ ,  $m \geq 2$  client campaigners, the revenue matrix  $A$ , a budget  $k_i$  on the seed set size for each campaigner  $C_i$ , that is,  $|S_i| = k_i$  for  $1 \leq i \leq m$ , and a positive integer  $R$ , can we find a seed set for each campaigner such that the expected revenue of the host is at least  $R$ ?

*Theorem 1:* Following the MCIC model of influence cascading, the decision version of Problem 1 is NP-hard.

*Proof:* We shall prove the NP-hardness by performing a reduction from the NP-complete set-cover problem. Let us consider an instance of the set-cover problem, defined by a collection of subsets  $S = \{S_1, S_2, \dots, S_r\}$  of a ground set  $U = \{u_1, u_2, \dots, u_n\}$ ; we wish to know whether there exist  $k$  of the subsets whose union is equal to  $U$ . Now, we consider another identical instance of the previous set cover problem, given by a collection of subsets  $S' = \{S'_1, S'_2, \dots, S'_r\}$  of the ground set  $U' = \{u'_1, u'_2, \dots, u'_n\}$ . We construct our revenue maximization problem for  $m = 2$  competing campaigners:  $C_1$  and  $C_2$ , as follows. For each  $u_i \in U$ ,  $u'_i \in U'$ ,  $S_i \in S$ , and  $S'_i \in S'$ , we include a node in the network. The two nodes corresponding to element pairs  $u_i, u'_i$  are connected by a bi-directed edge of probability 1. If a subset  $S_i$  covers an element  $u_j$ , we add a directed edge of probability 1 from node  $S_i$  to node  $u_j$  in the network. Analogously, if some subset  $S'_i$  covers an element  $u'_j$ , we also add a directed edge of probability 1 from node  $S'_i$  to node  $u'_j$  in the network. The revenue matrix has the following form:  $A_{1,u_i} = 1$  for all  $u_i \in U$ ,  $A_{2,u'_j} = 1$  for all  $u'_j \in U'$ , all other entries in the revenue matrix are 0. Finally, we also assume that there is a budget  $k$  on seed-set size for each campaigner. In this setting, there is a solution to our revenue maximization problem with the host's expected revenue at least  $2n$ , if and only if there is a solution to the set cover problem. Hence, the theorem. ■

*Theorem 2:* Following the K-LT model of influence cascading, the decision version of Problem 1 is NP-hard.

*Proof:* We prove the NP-hardness by performing a reduction from the NP-complete set cover problem, defined by a collection of subsets  $S = \{S_1, S_2, \dots, S_r\}$  of a ground set  $U = \{u_1, u_2, \dots, u_n\}$ ; and we want to know whether there exist  $k$  of the subsets whose union is equal to  $U$ . Now, we construct our revenue maximization problem with the K-LT model and for  $m = 2$  competing campaigners:  $C_1$  and  $C_2$ , as follows. For each element  $u_i \in U$ , there is a node in the network with activation threshold  $\frac{1}{r+1}$ . For each subset  $S_i \in S$ , we add two nodes  $v_i$  and  $v'_i$  in the network, each having an activation threshold 1. Now, if a subset  $S_i$  covers an element  $u_j$ , we add a directed edge of probability  $\frac{1}{2r}$  from node  $v_i$

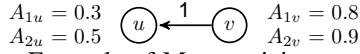


Fig. 4: Counter-Example of Monotonicity

to node  $u_j$ , and another directed edge of probability  $\frac{1}{2r}$  from node  $v'_i$  to node  $u_j$ . Note that the sum of the probabilities of all incoming edges to any node  $u_j$  is at most 1. The revenue matrix  $A$  has the following form:  $A_{1,u_i} = 1$  for all  $u_i \in U$ ,  $A_{2,u_i} = 1$  for all  $u_i \in U$ , and all other entries in  $A$  are 0. Finally, we also assume that there is a budget  $k$  on seed-set size for each campaigner. In this setting, there is a solution to our revenue maximization problem with the host's expected revenue at least  $n$ , if and only if there is a solution to the set cover problem. Hence, the theorem. ■

Unlike the classical viral marketing problem [13], our revenue maximization problem, under both MCIC and K-LT models, is *neither monotonic, nor sub-modular*. Therefore, an iterative greedy algorithm [13] that maximally increases the marginal gain at every iteration, and which has been widely-used to derive a solution with theoretical approximation bounds for the conventional viral marketing problem, can no longer be employed in our case for deriving similar approximation guarantees. Hence, we first design our novel approximated solutions in Sections IV and V, where we provide theoretical performance guarantees in the presence of some additional constraints. Finally, in Section VI, we also provide more efficient greedy solutions for our problem. However, prior to introducing our solution techniques, we demonstrate below the non-monotonicity and non-sub-modularity of the revenue maximization problem using counter-examples.

**Non-Monotonicity.** In Figure 4, we assume that node  $v$  is already assigned to  $S_2$  (i.e., seed set of campaigner  $C_2$ ) and that we still need to assign the seed set to campaigner  $C_1$ . Under both the MCIC and the K-LT model, node  $u$  will be activated by node  $v$  with probability 1. Therefore, the host's revenue is  $0.5 + 0.9 = 1.4$  with only node  $v$  assigned to the seed set  $S_2$  of campaigner  $C_2$ . If we now assign node  $u$  to the seed set  $S_1$  of campaigner  $C_1$ , the host's revenue is reduced to  $0.3 + 0.9 = 1.2$ . If we assign the seed sets in the other order (i.e., first  $S_1$  to  $C_1$  and then  $S_2$  to  $C_2$ ), then the revenue would have increased from  $0.3 + 0.8 = 1.1$  to  $0.3 + 0.9 = 1.2$ . In general, the revenue maximization problem is non-monotonic with respect to addition of seed sets.

**Non-Sub-Modularity.** Let us denote by  $F(S) = \sum_{i=1}^m \sum_{u \in V} [A_{iu} \cdot Pr(u, i, S)]$ . Here,  $S = \{S_1, S_2, \dots, S_m\}$  is the collection of  $m$  seed sets corresponding to  $m$  different campaigners. In order to illustrate non-sub-modularity, we need to show that the following inequality does not always hold:  $F(S \cup S_i) - F(S) \geq F(S' \cup S_i) - F(S')$ , where  $S' \supseteq S$ ,  $S_i \notin S'$ . In Figure 5, we assume that  $S : \{u_1 \text{ assigned to } C_1\}$ ; and  $S' : \{u_1 \text{ assigned to } C_1, u_2 \text{ assigned to } C_2\}$ . Under the MCIC model, node  $v$  is always activated, if either node  $u_1$  or  $u_2$  is activated. Under the K-LT model, assume we use an activation threshold of 0.4 for node  $v$ . Therefore, node  $v$  is always activated under the K-LT model as well. Now, we assign  $v$  to  $S_3$  (i.e., seed set of campaigner  $C_3$ ) and check the sub-modularity criteria for  $S$  and  $S'$ . We get  $F(S \cup S_3) - F(S) = (0.7 + 0.1) - (0.7 + 0.5) = -0.4$  and  $F(S' \cup S_3) - F(S') = (0.7 + 0.9 + 0.1) - (0.7 + 0.9 + (0.5 \times \frac{1}{2}) + (0.1 \times \frac{1}{2})) = -0.2$ . Therefore, the sub-modularity property is not satisfied.

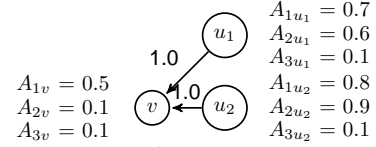


Fig. 5: Counter-Example of Sub-Modularity

#### IV. SOLUTION WITH INDEPENDENT CASCADE MODEL

In this section, we consider the revenue maximization problem under the MCIC influence cascading model. For simplicity, we assume that each campaigner has a seed set of the same size  $k$ .

**Overview.** Although our revenue maximization problem is NP-hard over graphs, we shall illustrate that the problem is solvable in polynomial time in a tree dataset. Therefore, we consider a two step heuristic approach: first, given a graph dataset, we extract the *most influential tree*, which will be formally defined in Section IV-C. Intuitively, the most influential tree approximates a social network by preserving the *most influential path* between every pair of nodes as much as possible [14]. A path between a source and a destination node is called the most influential path if the probability of influence cascading along that path is maximal in comparison with all other paths between these two nodes. While most-influential-path-based approaches were used in the past, e.g. [7], to solve the classical viral marketing problem, our current problem is a different one. We design a polynomial-time exact algorithm to solve the host's revenue maximization problem over a tree dataset (Section IV-A). We describe our algorithm with a simpler binary tree in the following section, and later in Section IV-B, we show how to convert a tree to an equivalent binary tree suitable for our method.

##### A. Exact Solution over Directed Binary Trees

On a directed tree, a node  $v$  can only be activated by its closest ancestor  $u$ , including the node itself, such that  $u$  belongs to one of the seed sets  $S_1, S_2, \dots, S_m$ . This is simply because  $u$  blocks the path from any farther ancestor  $u'$  to  $v$ , where  $u'$  is also a seed node. Therefore,  $Pr(v, i, S)$ , the probability that node  $v$  will be influenced by  $C_i$ 's campaign, has the following expression in case of a directed tree.

$$Pr(v, i, S) = \begin{cases} 0 & \text{if } u \notin S_i; \\ \prod_{(u', v') \in Path(u, v)} p_{u'v'} & \text{otherwise.} \end{cases}$$

Here,  $u$  denotes the closest ancestor of  $v$ , such that  $u$  is a seed node. First, we compute for every node  $v$  in the tree dataset, the probability that  $v$  gets activated by each of its ancestors  $u$ , which is simply  $\prod_{(u', v') \in Path(u, v)} p_{u'v'}$ . We store these activation probabilities in a table  $\mathcal{B}$  of size  $\mathcal{O}(nd)$ , where  $n$  is the total number nodes in the tree and  $d$  is the depth of the tree. Let  $\mathcal{B}(u, v)$  denote the activation probability that node  $v$  is activated by its ancestor  $u$ . Clearly,  $\mathcal{B}(v, v) = 1$ . The computation of table  $\mathcal{B}$  requires  $\mathcal{O}(nd)$  time, if we re-use the activation probabilities from a parent node in order to compute the activation probabilities for its children nodes.

Next, we apply a dynamic-programming-based algorithm to find the optimal seed sets for all campaigners over a directed binary tree. For this purpose, we introduce another table OPT

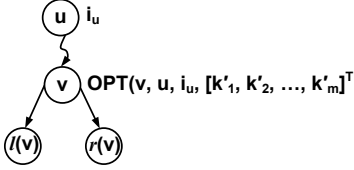


Fig. 6: Exact Solution over Binary Tree: MCIC model

of the form  $\text{OPT}(v, u, j, [k'_1, k'_2, \dots, k'_m]^T)$ , where: **(a)**  $v$  is any node in the tree, **(b)** node  $u$  denotes the nearest ancestor of node  $v$  such that  $u$  is a seed node, **(c)**  $j \in (1, m)$  denotes the campaigner  $C_j$  such that  $u$  is a seed node of campaigner  $C_j$ , and **(d)** each  $k'_i \leq k$  denotes the number of seed nodes already assigned to campaigner  $C_i$  in the subtree rooted at  $v$ . An entry in the  $\text{OPT}$  table, e.g.,  $\text{OPT}(v, u, j, [k'_1, k'_2, \dots, k'_m]^T)$  represents the host's expected revenue for the optimal assignment of all seed sets  $S_i$ ,  $i \in (1, m)$ ,  $|S_i| = k'_i$  over the subtree rooted at node  $v$ : given  $u$ , which is  $v$ 's nearest ancestor that is a seed node, is rather assigned as a seed node to campaigner  $C_j$ . It can be noted that the size of  $\text{OPT}$  table is  $\Theta(ndmk^m)$ . The entries in  $\text{OPT}$  are computed by performing a post-order traversal over the tree dataset as given in Equations 2, 3, 4.

In our dynamic programming,  $\text{OPT}(v, u, j, [k'_1, k'_2, \dots, k'_m]^T)$  is computed as the maximum over two cases. **(a)**  $\text{Case}_1$ : the first case considers the scenario when  $v$  is not selected as a seed node, and **(b)**  $\text{Case}_2$ : the second case considers the situation when  $v$  is assigned to some campaigner as a seed node. Here,  $l(v)$  and  $r(v)$  denote the left and right subtrees of  $v$ , respectively, as illustrated in Figure 6. For simplicity of description, we assumed that the budget of seed-set size for each of the  $m$  campaigners is  $k$ . Then, to fill one entry in the  $\text{OPT}$  table, we need  $\mathcal{O}(mk^m)$  time. Therefore, the time complexity of our dynamic programming is  $\mathcal{O}(ndm^2k^{2m})$ .

It is important to note that the dynamic programming terminates by computing the  $\text{OPT}$  entries for the root node  $v_r$ , that is,  $\text{OPT}(v_r, -1, i, [k, \dots, k]^T)$  for all  $i \in (1, m)$ . The value  $-1$  at the second index indicates that the root node  $v_r$  does not have an ancestor. Therefore,  $\text{OPT}(v_r, -1, i, [k, \dots, k]^T)$  is invariant of  $i$ . Once we terminate our dynamic programming, we determine whether a node will be a seed node (and if so, that node will be assigned to which campaigner) by *backtracking* using the  $\text{OPT}$  entries of its children nodes. The backtracking process requires another  $\mathcal{O}(mnk^m)$  time. Therefore, the overall time complexity of finding the optimal seed nodes over a directed binary tree is  $\mathcal{O}(ndm^2k^{2m})$ . We note that our dynamic-programming-based exact solution over directed binary trees is polynomial-time in the number of tree nodes; however, it has exponential time-complexity in the number of client campaigners.

**Space and Time Complexity.** In this section, we summarize the space and time complexity of our dynamic-programming-based solution. The space complexity is  $\mathcal{O}(ndmk^m)$ : table  $\mathcal{B}$  has size  $\mathcal{O}(nd)$  and table  $\text{OPT}$  has size  $\Theta(ndmk^m)$ . The time complexity of our dynamic programming is  $\mathcal{O}(ndm^2k^{2m})$ .

### B. Directed Trees to Binary Trees Conversion

Our dynamic-programming-based exact solution for the revenue maximization problem (Problem 1) can be applied over non-binary trees. In fact, given a directed tree, we first convert it to an equivalent directed binary tree. We use the conversion technique in [15]. For each non-leaf node  $v$  with

children  $v_1, v_2, \dots, v_\Delta$ , where  $\Delta > 2$  in the original tree, we replace  $v$  with a binary tree of depth at most  $\log \Delta$  and leaves  $v_1, v_2, \dots, v_\Delta$ . For each newly introduced node  $u$ , we assign the revenue  $A_{iu} = 0$ , for all campaigners  $i \in (1, m)$ . While applying our dynamic programming (Section IV-A), we also incur an additional constraint that no such newly introduced node can be selected as a seed node. Finally, for each newly introduced edge, the direction is always from the root towards the leaves, and each of them has probability 1. The incoming edges to the leaves  $v_1, v_2, \dots, v_\Delta$  have the same probability as that of the previous incoming edges to nodes  $v_1, v_2, \dots, v_\Delta$ , respectively. This conversion process ensures that the newly introduced edges and nodes will not affect the MCIC propagation model as in the original directed tree.

As shown in [15], for the aforementioned tree-to-binary-tree conversion method, the number of nodes in the equivalent binary tree is at most twice the number of nodes in the directed input tree, and the depth of the binary tree is at most a factor of  $\log \Delta^*$  larger than the depth of the original tree, where  $\Delta^*$  is the maximum out-degree of any node in the input tree.

$$\text{OPT} \left( v, u, j, \begin{bmatrix} k'_1 \\ \vdots \\ k'_m \end{bmatrix} \right) = \max\{\text{Case}_1, \text{Case}_2\} \quad (2)$$

$$\begin{aligned} \text{Case}_1 = & \max_{k'_1=0}^{k'_1} \dots \max_{k'_m=0}^{k'_m} \left\{ \text{OPT} \left( l(v), u, j, \begin{bmatrix} k'_1 \\ \vdots \\ k'_m \end{bmatrix} \right) \right. \\ & \left. + \text{OPT} \left( r(v), u, j, \begin{bmatrix} k'_1 - k''_1 \\ \vdots \\ k'_m - k''_m \end{bmatrix} \right) + A_{j,v} \times \mathcal{B}(u, v) \right\} \quad (3) \end{aligned}$$

$$\begin{aligned} \text{Case}_2 = & \max_{i=1}^m \left\{ \max_{k'_1=0}^{k'_1} \dots \max_{k'_i-1}^{k'_i-1} \dots \max_{k'_m=0}^{k'_m} \left\{ \text{OPT} \left( l(v), v, i, \begin{bmatrix} k''_1 \\ \vdots \\ k''_i \\ \vdots \\ k''_m \end{bmatrix} \right) \right. \right. \\ & \left. \left. + \text{OPT} \left( r(v), v, i, \begin{bmatrix} k'_1 - k''_1 \\ \vdots \\ k'_1 - k''_i - 1 \\ \vdots \\ k'_m - k''_m \end{bmatrix} \right) + A_{i,v} \right\} \right\} \quad (4) \end{aligned}$$

### C. Graphs to Most Influential Directed Tree Extraction

The revenue maximization problem is  $\text{NP}$ -hard in directed graphs (Theorem 2). Therefore, given a directed and connected graph  $G$ , we first extract the *most influential tree*  $T^*$ , which is a directed spanning tree of  $G$ , and formally defined below.

**Definition 1 (Most Influential Tree):** Given a connected graph  $G = (V, E, P)$  with a root node  $v_r$ , the most influential tree  $T^* = (V, E_{T^*}, P)$  with  $E_{T^*} \subseteq E$  is a directed spanning tree of  $G$ , with the same root node  $v_r$ , such that the product of the edge probabilities in  $T^*$  is maximized. Formally,

$$T^* = \arg \max_{T \in \text{SpanningTrees}(G)} \prod_{(u,v) \in E_T} p_{u,v} \quad (5)$$

$\text{SpanningTrees}(G)$  denotes all directed spanning trees of  $G$ . Intuitively, the most influential tree aims at preserving the most influential path between every pair of nodes as much as possible. These most influential paths play an important role in influence cascade over real-world social networks [14].

The problem of finding the most influential tree can be converted to the problem of finding the minimum-cost *directed* spanning tree by minimizing the sum of negative logarithms to the edge probabilities in  $T^*$  as given in Equation 6.

$$T^* = \arg \min_{T \in \text{SpanningTrees}(G)} \sum_{(u,v) \in E_T} -\log(p_{u,v}) \quad (6)$$

Thus, one can find the most influential directed tree in time  $\mathcal{O}(e+n \log n)$  due to Gabow et al. [10]. It is important to note that [10] requires some root node  $v_r$  to be present in the input graph  $G$  such that all other nodes in  $G$  are reachable from  $v_r$ . Therefore, we first add a dummy root node  $v_r$  and then connect all nodes in  $G$  to  $v_r$ , with edges directed towards the nodes in  $G$ . Each of these newly-introduced edges is assigned a very low edge-probability. For the dummy root node  $v_r$ , we also assign revenue  $A_{iv_r} = 0$  for all campaigners  $i \in (1, m)$ ; and then, we further incur an additional constraint that  $v_r$  cannot be selected as a seed node during our dynamic-programming-based exact solution over the most influential tree  $T^*$ .

## V. SOLUTION WITH LINEAR THRESHOLD MODEL

In this section, we consider the revenue maximization problem (Problem 1) under the K-LT influence cascade model. We recall that our problem is NP-hard under the K-LT model (Theorem 2). However, we shall later illustrate that given an *already-selected* set of seed nodes, one can optimally partition these seed nodes among  $m$  campaigners in polynomial time such that the host's expected revenue is maximized. Therefore, we design a two-step heuristic technique to solve our original revenue maximization problem with a *theoretical performance guarantee* of  $\frac{1}{m}(1-\frac{1}{e})$ , where  $m$  is the number of campaigners.

**Overview.** In the first phase (Section V-A), the host *optimistically* assumes that it is possible to influence each user by a campaign that gives the maximum revenue to the host for that user, considering all other campaigns. This is equivalent to assigning, for each user  $u$ , a revenue  $A_u$  which is the maximum of  $A_{iu}$  values over all campaigners  $i$ . Thus, the host identifies  $mk$  seed nodes assuming there is only *one campaigner* and with the objective that her expected revenue is maximized under this optimistic assumption. However, we recall that in reality, there are  $m$  campaigners, each with a seed-set of size  $k$ . Therefore, in the second step (Section V-B), the host partitions these *already-selected*  $mk$  seeds among  $m$  campaigners with the objective that her expected revenue is maximized under the actual *multi-campaigner* setting and considering the original revenue matrix. Below, we describe both these steps in details. For simplicity, we assume that the budget of seed-set size for each of the  $m$  campaigners is  $k$ .

### A. Optimistic Seed Set Selection

In the first phase, the host optimistically assumes that each user in the network can be influenced by a campaign such that the corresponding campaigner gives the maximum amount of

money for that particular user. In other words, for each user  $u$  in the network, the host optimistically assigns a revenue  $A_u$  which is the maximum of  $A_{iu}$  values over all campaigners  $i$ . Formally,  $A_u = \max_{i \in (1, m)} \{A_{iu}\}$ . Therefore, the host solves the following problem in the first step.

*Problem 2 (Optimistic Seed Set Selection):* Assuming there is only one campaigner and given a revenue  $A_u$  for each user  $u$  in the network, find the seed set of size  $mk$  such that the expected revenue of the host is maximized. Formally,

$$\arg \max_S \sum_{u \in V} [A_u \cdot Pr_{LT}(u, S)]$$

such that  $|S| = mk$  (7)

Here,  $Pr_{LT}(u, S)$  denotes the expected spread of an influence from the seed set  $S$  to node  $u$  following the classic Linear threshold (LT) model with one campaigner. Unfortunately, Problem 2 is also NP-hard following [13]; nevertheless, the objective function is monotonic and sub-modular as shown in Theorem 3.

*Theorem 3:* The objective function of Problem 2 is sub-modular. Formally, let  $F(S) = \sum_{u \in V} [A_u \cdot Pr_{LT}(u, S)]$ . Then,

$$F(S \cup \{v\}) - F(S) \geq F(S_1 \cup \{v\}) - F(S_1) \quad (8)$$

Here,  $S_1 \supseteq S$  and  $v \notin S_1$ .

*Proof:* The proof follows by considering the *live-edge* model, which is shown to be equivalent to the LT model in [13]. In the live-edge model, each node  $v$  picks at most one of its incoming edges at random, that is, it selects the incoming edge from  $u$  with probability  $p_{u,v}$ , and it does not select any incoming edge with probability  $1 - \sum_{u \in in(v)} p_{u,v}$ . Let  $X$  be one possible world with probability  $Prob(X)$  under the live-edge model, and  $R_X(S)$  be the host's revenue due to nodes that are reachable from the seed set  $S$  in that possible world  $X$ . One may verify that  $R_X(S)$  is sub-modular with respect to  $S$ . Now, our objective function  $F(S)$  is given by:

$$F(S) = \sum_{\text{all possible world } X} [Prob(X) \cdot R_X(S)] \quad (9)$$

As the non-negative linear combination of sub-modular functions is also sub-modular,  $F(S)$  is sub-modular.  $\blacksquare$

Thus, we apply an iterative hill-climbing algorithm (Algorithm 1) that finds the seed set with an approximation guarantee  $(1 - \frac{1}{e})$  of the optimal solution [20]. The hill-climbing algorithm works in  $mk$  iterative steps. At each iteration, the algorithm selects a non-seed node  $u$  as a seed node, such that the expected revenue due to  $u$  and the previously selected seed nodes is maximized. Our hill-climbing-based iterative solution for the optimistic seed selection problem (Problem 2) is similar to state-of-the-art viral marketing techniques that identify the top- $k$  seed nodes for a single campaigner such that its expected influence spread in the network is maximized [13]. Although we optimize the host's expected revenue instead of the expected influence spread, due to the single-campaigner and sub-modular nature of Problem 2, one can easily apply an existing viral marketing algorithm [7], [11], [13], [16] (with some modification in the objective function) as the underlying technique to solve Problem 2.



---

**Algorithm 1** Hill-Climbing for Optimistic Seed Set Selection
 

---

**Require:** Graph  $G = (V, E, P)$ ,  $A_u = \max_i \{A_{iu}\} \forall u \in V$

**Ensure:** Seed set  $S$  of size  $mk$

- 1:  $S = \phi$
  - 2: **for**  $i = 1$  **to**  $mk$  **do**
  - 3:    $v = \arg \max_{v \in V \setminus S} F(S \cup \{v\})$    //  $F()$  is defined in Th. 3
  - 4:    $S = S \cup \{v\}$
  - 5: **end for**
  - 6: Output  $S$
- 

We shall later show in Theorem 4 that the iterative hill-climbing algorithm for the optimistic seed selection, coupled with an optimal partition of those seed sets, generates a solution to the original revenue maximization problem with the approximation guarantee  $\frac{1}{m}(1 - \frac{1}{e})$ , where  $m$  is the number of campaigners.

**Time Complexity.** The time complexity of our iterative hill climbing algorithm is  $\mathcal{O}(mkn(n+e)t)$ , where  $mk$  is the total number of seed nodes identified, and  $t$  is the number of Monte-Carlo samples performed over the entire graph in order to find one seed node.

### B. Partition of Seed Set

In the second phase, the host optimally partitions the previously selected  $mk$  seed nodes among  $m$  campaigners, such that her expected revenue is maximized under the actual *multi-campaigner* K-LT model and considering the original revenue matrix. We formally define our problem statement for the second step as follows.

*Problem 3 (Optimal Seed-Set-Partition):* Given already selected seed set  $S$  of size  $mk$  and the revenue matrix  $(A_{iu})_{m \times n}$ , partition  $S$  into  $m$  subsets  $S_1, S_2, \dots, S_m$ , such that each  $S_i$  has size  $k$ , and the expected revenue of the host is maximized following the multi-campaigner K-LT model.

We show that Problem 3 can be solved optimally in polynomial time using a dynamic-programming-based approach. For this purpose, we introduce the notion of *individual revenue* of the host from every seed node.

*Definition 2 (Individual Revenue):* The *individual revenue*  $\mathcal{R}_i(u)$  represents the expected revenue of the host from a seed node  $u \in S$  when  $u$  is assigned to the  $i$ -th campaigner  $C_i$ .

**Individual Revenue Computation.** We now describe our method to compute individual revenues. We start by randomly assigning a distinct number from 1 to  $mk$  to every seed node in  $S$ . Let us denote by  $I(u)$  the number assigned to seed node  $u$ . We also associate a list  $\mathcal{L}$  of size  $mk$  with each node  $v$  in the network. The  $j$ -th entry of list  $\mathcal{L}(v)$ , denoted as  $\mathcal{L}_j(v)$ , represents the spread that some seed node  $u \in S$  can achieve at node  $v$  following the K-LT model, where  $I(u) = j$ . For a seed node  $u \in S$ , we initialize:

$$\mathcal{L}_j(u) = \begin{cases} 1, & \text{if } I(u) = j; \\ 0, & \text{otherwise.} \end{cases}$$

Next, we simulate the K-LT model starting from seed nodes in  $S$ . At any discrete step of the K-LT model, if some node  $v$  becomes active, we consider all its in-neighbors  $v' \in in(v)$

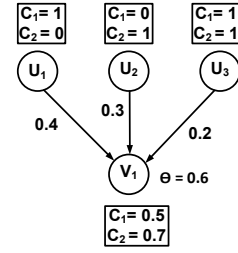


Fig. 7: Example of Individual Revenue Computation

that were activated in the previous step. We compute  $\mathcal{L}_j(v)$  as follows:

$$\mathcal{L}_j(v) = \frac{\sum_{\substack{v' \in in(v) \\ v' \text{ activated in prev. step}}} [p_{v'v} \times \mathcal{L}_j(v')]}{\sum_{\substack{v' \in in(v) \\ v' \text{ activated in prev. step}}} p_{v'v}} \quad (10)$$

Finally, we compute the individual revenue  $\mathcal{R}_i(u)$  for every seed node  $u \in S$  and for every campaigner  $C_i$  as given in Equation 11.

$$\mathcal{R}_i(u) = \sum_{v \in V} [A_{i,v} \times \mathcal{L}_{I(u)}(v)] \quad (11)$$

We refer to  $\mathcal{R}_i(u)$  as the individual revenue of the host due to seed node  $u$ , when  $u$  is assigned to campaigner  $C_i$ . We demonstrate the computation of individual revenues with an example below.

*Example 4:* In Figure 7, we assume there are three seed nodes:  $u_1, u_2$ , and  $u_3$ , and also two campaigners:  $C_1$  and  $C_2$ . The seed nodes are not assigned to any specific campaigners yet. We show the revenue vectors corresponding to each node inside the rectangular boxes. In the beginning, all seed nodes are activated, and in the next round,  $v_1$  gets activated, since its activation threshold  $\theta_{v_1} = 0.6 < p_{u_1,v_1} + p_{u_2,v_1} + p_{u_3,v_1} = 0.9$ . Following Equation 10, we get:  $\mathcal{L}_{I(u_1)}(v_1) = \frac{0.4}{0.4+0.3+0.2} = \frac{4}{9}$ . Similarly,  $\mathcal{L}_{I(u_2)}(v_1) = \frac{3}{9}$ , and  $\mathcal{L}_{I(u_3)}(v_1) = \frac{2}{9}$ . Finally, we compute the individual revenues by following Equation 11. For example,  $\mathcal{R}_1(u_1) = \sum_{v=u_1, u_2, u_3, v_1} A_{1,v} \times \mathcal{L}_{I(u_1)}(v) = 1 \times 1 + 0 + 0 + 0.5 \times \frac{4}{9} = 1.22$ . Similarly, we have:  $\mathcal{R}_1(u_2) = 0 + 0 + 0 + 0.5 \times \frac{3}{9} = 0.17$ .

**Properties of Individual Revenue.** The individual revenue  $\mathcal{R}_i(u)$  satisfies several interesting properties which are critical for our dynamic-programming-based exact solution.

*Proposition 1:* For a given seed node  $u$  and a given campaigner  $C_i$ ,  $1 \leq i \leq m$ ,  $\mathcal{R}_i(u)$  is invariant to how other seed nodes are assigned to the various campaigners.

*Proposition 2:* If some seed nodes  $u_1, u_2, \dots, u_j \in S$  are all assigned to a given campaigner  $C_i$ ,  $1 \leq i \leq m$ , then the expected revenue of the host due to  $u_1, u_2, \dots, u_j$  is simply the aggregation of  $\mathcal{R}_i(u_1), \mathcal{R}_i(u_2), \dots, \mathcal{R}_i(u_j)$ ; and this is invariant to how the remaining seed nodes are assigned to other campaigners.

We omit the proofs due to limitation of space. The first proposition directly follows from the definition of the K-LT model — given a pre-defined seed set  $S$ , the activation of



other nodes in the network is determined by the first phase of the K-LT model, that is, the classical LT model assuming all the campaigners are cascading the same information. More specifically, whether a node in the network will be activated or not is independent of how  $S$  is partitioned among multiple campaigners [19]. The partition of  $S$  only influences the following: among the active nodes in the network, which one will adopt what campaign and with how much probability. The second proposition follows from the linearity property of  $\mathcal{L}_j(v)$  in Equation 10, that is,  $\mathcal{L}_{\{u_1, u_2\}}(v) = \mathcal{L}_{u_1}(v) + \mathcal{L}_{u_2}(v)$ , for any two distinct seed nodes  $u_1, u_2 \in S$  and for any node  $v$  in the graph.

*Example 5:* In Figure 7,  $\mathcal{R}_1(u_1) = 1.22$  and  $\mathcal{R}_1(u_2) = 0.17$ . Note that,  $\mathcal{R}_1(u_1) + \mathcal{R}_1(u_2) = 1.39$ , and this is exactly same as  $\mathcal{R}_1(\{u_1, u_2\})$ , that is, the expected revenue of the host when both  $u_1$  and  $u_2$  are assigned to the campaigner  $C_1$ .

**Dynamic Programming Based Exact Solution.** We are now ready to describe our dynamic-programming-based exact algorithm to solve the optimal seed-set-partitioning problem (Problem 3). Our algorithm processes the seed nodes  $u \in S$  in the ascending order of their assigned  $I(u)$  numbers. We recall that  $I(u) \in (1, mk)$ . The dynamic programming maintains a table **EXACT** $(j, [k'_1, k'_2, \dots, k'_m]^T)$ , that stores the optimal expected revenue of the host when we have already partitioned the seed nodes with number from 1 to  $j$  into  $m$  subsets  $\{S_1, S_2, \dots, S_m\}$ , such that  $|S_i| = k'_i \leq k$ , and we have also assigned them to the respective campaigners. Clearly,  $1 \leq j \leq mk$  and  $j = \sum_{i=1}^m k'_i$ . The dynamic programming proceeds as given in Equation 12.

In Equation 12,  $u$  denotes the node with  $I(u) = j$ . The optimal assignment of the last seed node is determined by **EXACT** $(mk, [k, k, \dots, k]^T)$ . The optimal assignment of previous seed nodes are determined by *backtracking* with the usage of **EXACT** values. The correctness of our dynamic-programming-based solution follows from Propositions 1, 2.

$$\text{EXACT} \left( j, \begin{bmatrix} k'_1 \\ k'_2 \\ \vdots \\ k'_m \end{bmatrix} \right) = \max_{i \in (1, m)} \left\{ \text{EXACT} \left( j-1, \begin{bmatrix} k'_1 \\ \vdots \\ k'_i - 1 \\ \vdots \\ k'_m \end{bmatrix} \right) + \mathcal{R}_i(u) \right\} \quad (12)$$

**Space and Time Complexity.** Our algorithm has space complexity  $\mathcal{O}(k^m)$  due to the **EXACT** table. The time complexity of our dynamic programming is  $\mathcal{O}(mk^m)$ . This is because we need to fill the table of size  $k^m$ ; and in order to fill each entry in the table, we compute the maximum of  $m$  values. The backtracking requires another  $\mathcal{O}(m^2k)$  time; since there are  $mk$  seed nodes that we need to assign to different campaigners; and for each seed node, we require to compare  $m$  values to find the best assignment. In addition, one needs to compute the expected revenue vectors for all seed nodes by running a breadth-first-search from each of these  $mk$  seed nodes. Hence, the time required to compute the expected

revenue vectors for all seed nodes is  $\mathcal{O}(mk(n+e))$ , where  $n$  and  $e$  are the number of nodes and edges in the graph, respectively. Therefore, the time complexity of our exact solution is  $\mathcal{O}(mkn + mke + m^2k + mk^m)$ . We note that our optimal seed-set-partitioning solution requires polynomial time in the size of the graph.

**Performance Guarantee.** Theorem 4 provides the overall *approximation guarantee* of our method for the host's revenue maximization problem under the K-LT model.

*Theorem 4:* The iterative hill-climbing solution of the optimistic seed selection (Problem 2), coupled with the *optimal* partition of those seed sets (Problem 3), guarantees  $\frac{1}{m}(1 - \frac{1}{e})$  approximation to the original revenue maximization problem (Problem 1) under the K-LT model, and with the assumption that each campaigner has the same number of seed nodes. Here,  $m$  is the number of campaigners.

*Proof:* See Appendix. ■

### C. K-LT Model over Tree: An Exact Solution

In this section, we show that the revenue maximization problem under the K-LT model can be solved optimally in polynomial time over a tree dataset. Our proof is based on a reduction of the current problem into an equivalent problem of revenue maximization under the MCIC model over a tree dataset, which can be solved optimally in polynomial time as illustrated earlier in Section IV. Our reduction works as follows. Since each node in a tree has at most one incoming edge, we eliminate those incoming edges for which the probability on the edge is less than the activation threshold of the destination node. On the other hand, we retain those incoming edges for which the edge-probability is higher than or equal to the activation threshold of the destination node, and we reassign a probability 1 to all these retained edges. One may note that all the retained edges are also independent to each other even under the K-LT model, as each node has at most one retained incoming edge in a tree. Therefore, we can apply the dynamic-programming-based exact solution for the MCIC model to find an exact solution in polynomial time in the size of the tree dataset.

## VI. GREEDY SOLUTIONS

Our proposed techniques in the previous sections are polynomial-time with respect to the graph size, and they also provide theoretical performance guarantees under additional constraints, e.g., exact solution over tree datasets for both MCIC and K-LT models, and  $\frac{1}{m}(1 - \frac{1}{e})$ -optimal solution over any graph under K-LT model. Nevertheless, the running time of our algorithms increases exponentially with the number of seed nodes. Therefore, in this section, we propose more efficient greedy techniques for the host's revenue maximization problem. For ease of discussion, we refer to our earlier solution techniques in Sections IV and V as RevMax-Combined (RevMax-C), while we call our greedy solutions as RevMax-Separate (RevMax-S).

**RevMax-Separate.** This is a greedy method as given in Algorithm 2. We first sort the campaigners in descending order of  $\sum_{u \in V} A_{iu}$ , that is, the aggregated money that each campaigner  $C_i$  is willing to provide to the host if all the users in the

---

**Algorithm 2** RevMax-Separate: Greedy Seed Set Selection

---

**Require:** Graph  $G = (V, E, P)$ , revenue matrix  $(A_{iu})$ ,  $m$  campaigners

**Ensure:** Seed sets  $S_1, \dots, S_m$ , each of size  $k$

```
1: Sort and process campaigners in descending order of  $\sum_{u \in V} A_{iu}$ 
   for each campaigner  $C_i; 1 \leq i \leq m$ 
2: for  $i = 1$  to  $m$  do
3:    $S_i = \phi$ 
4:   for  $j = 1$  to  $k$  do
5:      $v = \arg \max_{v \in V \setminus S_i} F_i(S_i \cup \{v\})$  2
6:      $S_i = S_i \cup \{v\}$ 
7:   end for
8:    $V = V \setminus S_i$ 
9: end for
10: Output  $S_1, \dots, S_m$ 
```

---

network adopt her product. Next, we process the campaigners in that sorted order. For each campaigner, we identify the top- $k$  seed nodes such that the host’s revenue is maximized by considering only that campaigner (and disregarding the existence of other campaigners). Nevertheless, in order to eliminate the influence-cascading effect of already-selected seed nodes of previous campaigners, we delete these already-selected seed nodes from the graph before identifying the top- $k$  seed nodes for the next campaigner (Line 8, Algorithm 2).

**Time Complexity.** The time complexity of our greedy algorithm is  $\mathcal{O}(mkn(n + e)t)$ , where  $m$  is the number of campaigners,  $k$  the number of seed nodes per campaigner,  $n$  and  $e$  are the number of nodes and edges in the graph, respectively, and  $t$  is the number of Monte-Carlo samples performed to find one seed node. We note that unlike our approximated algorithms in Sections IV and V, our greedy approach is very scalable — the running time increases linearly with the number of campaigners, number of seeds per campaigner, and polynomially with the size of the graph.

## VII. EXPERIMENTAL RESULTS

We present experimental results which illustrate the effectiveness, efficiency, and scalability of our revenue maximization algorithms. The code is implemented in C++ and the experiments were performed on a single core of a 132GB, 2.26GHz Xeon server.

### A. Experiment Setup

□ **Datasets:** We summarize our data sets in Table I. Additional results over tree datasets can be found in the Appendix.

TABLE I: Dataset Characteristics

Dataset	# Nodes	# Edges	Edge Prob: Mean, SD, Quartiles
<i>Flickr</i>	78 322	20 343 018	$0.09 \pm 0.06, \{0.06, 0.07, 0.09\}$
<i>DBLP</i>	684 911	4 569 982	$0.08 \pm 0.07, \{0.05, 0.05, 0.10\}$
<i>NetHEPT</i>	15 229	62 752	$0.28 \pm 0.28, \{0.0006, 0.27, 0.53\}$

**Flickr** (<http://www.flickr.com>). Flickr is an online community, where users share photos, and participate in common-interest groups. We borrowed the dataset from [21], where the probability of an edge between any two users is computed assuming *homophily*; in particular, the Jaccard coefficient of the interest groups that the two users belong to.

---

<sup>2</sup> $F_i(S_i) = \sum_{u \in V} [A_{iu} \cdot Pr(u, S_i)]$ , i.e., host’s revenue considering only campaigner  $C_i$ , with seed set  $S_i$

**DBLP** (<http://www.informatik.uni-trier.de/~ley/db/>). The dataset is a subset of the popular co-authorship network used in [21]. Here, the edge probabilities express the strength of the collaboration between the two incident authors. Particularly, in [21], the probabilities derive from an exponential cdf to the number of collaborations; hence, if two authors collaborated  $c$  times, we assign the corresponding probability as  $1 - \exp^{-c/10}$ .

**NetHEPT** (<http://www.arXiv.org>). This graph is created from the “High Energy Physics - Theory” section of the arXiv with papers from 1991 to 2003 [7]. Since there is no edge probabilities on this graph dataset, we synthetically assign probabilities on edges that simulates the community structure in a social network. We identify 60 non-overlapping communities from this graph dataset, each with 170 nodes. If an edge is completely inside a community, we uniformly assign a probability between 0.2 to 0.8; all other edges are assigned probabilities uniformly from 0 to 0.001. Such an edge probability assignment reflects the fact that users inside the same community usually have higher influence on each other than on someone else outside that community. The edge probabilities are assigned differently in both directions, i.e.,  $p_{uv} \neq p_{vu}$ .

For the K-LT model, in all datasets, if the sum of probabilities of all incoming edges to a node is more than 1, we normalize those edge probabilities by their aggregate value, such that the sum of probabilities for in-edges to every node is no more than 1 [21]. We also limit the number of Monte-Carlo samples to 1 000 in all our experiments [21].

□ **Number of Campaigners and Seed Nodes:** We vary the number of campaigners from 2 to 10, while the number of seed nodes per campaigner is varied from 5 to 100. We also consider the case when different campaigners allow a different number of seed nodes. In Appendix, we show additional experiments with  $m = 1$  campaigner, and our results attest that both RevMax-C and RevMax-S converge to a traditional influence maximization solution with a single campaigner.

□ **Revenue Distribution:** We consider five wide categories of revenue distribution in order to simulate various real-world scenarios.

**Uniform (U).** In this setting, each campaigner selects its target users uniformly over the network and independent of other campaigners. Therefore, in our framework, we assign every revenue-matrix-element  $A_{iu} = 1$  monetary unit, with probability  $\frac{1}{m}$ ; and  $A_{iu} = 0.1$  monetary unit, with probability  $(1 - \frac{1}{m})$ . Here,  $m$  is the number of campaigners. One may note that we have normalized the amount of money that a campaigner gives to the host for one user on a scale from 0.1 to 1 monetary units.

**Not Equal (NE).** In this setting, we consider only two campaigners — one campaigner  $C_1$  assigns a fixed revenue  $A_{1u} = 1$  for each user  $u$ , while the other campaigner  $C_2$  assigns a fixed revenue  $A_{2u} = 0.5$  for each user.

**Clustering with Low Competition (CLC).** In this setting, we assume that each campaigner’s target users form certain clusters in the network. In addition, we also assume that there are some users who belong to target sets of all the campaigners.

We call this model “clustering with low competition” as we limit the ratio of such mutually overlapping target users to a relatively small percentage. We simulate this setting as follows. We first partition the graph into 15 non-overlapping and highly-connected clusters, each cluster having equal number of nodes. For the first 5 clusters, all  $A_{iu}$  values are set to 1 monetary unit, i.e., 33% of the nodes belong to the target users of all campaigners. The remaining clusters are assigned to the campaigners in a round-robin manner. If a cluster is assigned to campaigner  $C_j$  as its target set, we then assign each  $A_{ju} = 0.5$  monetary unit, and the remaining  $A_{iu} = 0.1$  monetary unit, for all  $j \neq i$ , inside that cluster.

**Clustering with High Competition (CHC).** This setting is similar to the previous CLC setting — the only difference is that there is a relatively large number of users who belong to the target sets of all campaigners. We simulate this setting as before; however, we assign the first 10 out of the 15 clusters as the target sets for all campaigners. This implies that 66% of the nodes belong to the target users of all campaigners.

**Clustering with Not-Equal Competition (CNC).** This setting is also similar to the CLC setting, except we consider only two campaigners  $C_1$  and  $C_2$ ; and while the three possible  $A_{1u}$  values remain (1, 0.5, 0.1) as before, the three possible  $A_{2u}$  values are reduced to (0.5, 0.25, 0.05).

While U, CLC, and CHC models ensure almost equal host’s revenue from each of her client campaigners; for NE and CNC models, the host’s aggregated revenue from campaigner  $C_1$  is more than that from campaigner  $C_2$  considering all the network users.

□ **Comparing Methods:** We compare our approximated algorithms RevMax-C (Sections IV and V) and heuristic method RevMax-S (Section VI) with a randomized seed selection approach.

**Random.** We randomly select a distinct seed set for each campaigner. In our experiments, we did 10 runs of the Random method, and selected the best one that results in the maximum revenue out of all these 10 runs.

We compare the three aforementioned techniques — RevMax-C, RevMax-S, and Random under both IC and LT models. As the underlying viral marketing method in RevMax-S and RevMax-C, we use the CELF++ algorithm [11] due to its efficiency. We use the publicly-available source code of CELF++ provided by the respective authors [11].

□ **Evaluation Metrics:** We compare the host’s revenues obtained from RevMax-S and RevMax-C with that of the Random.

**Revenue Improvement Rate (RIR).** This is defined as the ratio of the host’s expected revenue obtained from the seed sets identified by RevMax-C (or, RevMax-S) with respect to the host’s revenue obtained from a random selection of seed sets.

### B. Performance: Effectiveness & Efficiency

We first demonstrate our results over the MCIC model (Section VII-B1), and the K-LT model (Section VII-B2). We also show our results with uneven revenue distribution (i.e., NE and CNC) in Section VII-B3. Since RevMax-C does not

TABLE II: Revenue Improvement Rate (RIR), MCIC Influence Cascading Model with 2 Campaigners, *NetHEPT* Dataset

Revenue Distribution	#Seed Nodes per Camp.	RIR RevMax-S	RIR RevMax-C
CRH	5	2.52	3.14
	10	2.92	3.28
	15	2.68	3.07
	20	1.94	2.23
CRL	5	3.30	3.33
	10	2.91	3.20
	15	2.48	2.94
	20	2.09	2.37
U	5	3.23	3.52
	10	2.12	2.04
	15	2.72	2.80
	20	2.34	2.52

TABLE III: Revenue Improvement Rate (RIR), MCIC Influence Cascading Model with 2 Campaigners, and 5 Seeds/Campaigner

Dataset	Revenue Distribution	RIR RevMax-S	RIR RevMax-C
NetHEPT	CRH	2.52	3.14
	CHL	3.30	3.33
	U	3.23	3.52
DBLP	CRH	1.04	1.04
	CRL	1.02	1.03
	U	1.03	1.03
Flickr	CRH	1.43	1.66
	CHL	1.20	1.15
	U	1.01	1.11

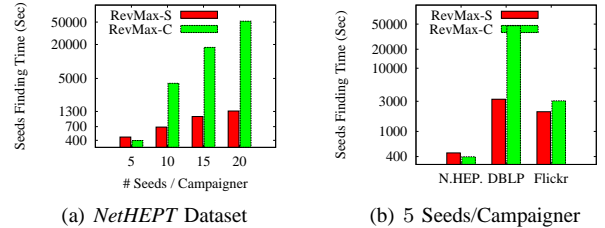


Fig. 8: Seed Sets Finding Time, MCIC Influence Cascading Model with 2 Campaigners

scale well with many campaigners and with a large number of seed nodes, we consider at most 5 campaigners and up to 20 seeds per campaigner in these experiments. The scalability of RevMax-S with more campaigners and seed nodes is illustrated later in Section VII-C.

1) *Performance with MCIC Model:* We present the revenue improvement rates with the MCIC model in Tables II and III. We find that the host’s revenue by following RevMax-C technique almost always outperforms that from the RevMax-S approach by a margin of 5%~10%. We show the corresponding efficiency results for the MCIC model in Figures 8(a) and 8(b). We find that RevMax-C requires less amount of time to identify the top- $k$  seed nodes as compared to that of RevMax-S, over the smaller *NetHEPT* dataset and for 5 seeds per campaigner. However, as we consider larger datasets and more seeds per campaigner, RevMax-C requires more time. This is because RevMax-C identifies the seed sets over the most influential tree of the corresponding graph dataset in an exact manner, and this process requires time  $\mathcal{O}(ndm^2k^{2m})$ . Clearly, the running time of RevMax-C increases at a higher rate as one increases the number of nodes  $n$ , and the number of seeds  $k$  per campaigner.

TABLE IV: Revenue Improvement Rate (RIR), K-LT Influence Cascading Model, CRH Revenue Distribution, *NetHEPT*

# Camp.	#Seed Nodes per Camp.	RIR	
		RevMax-S	RevMax-C
2	5	8.99	<b>9.20</b>
	10	7.73	<b>7.97</b>
	15	<b>6.89</b>	6.82
3	5	5.86	<b>5.94</b>
	10	7.01	<b>7.29</b>
	15	<b>6.10</b>	5.73
5	5	5.70	<b>5.85</b>
	10	<b>5.00</b>	4.83
	15	<b>5.04</b>	4.77

TABLE V: Revenue Improvement Rate (RIR), K-LT Influence Cascading Model with 2 Campaigners, *NetHEPT* Dataset

Revenue Distribution	#Seed Nodes per Camp.	RIR	
		RevMax-S	RevMax-C
CRH	5	8.99	<b>9.20</b>
	10	7.73	<b>7.97</b>
	15	<b>6.89</b>	6.82
CRL	5	8.62	<b>8.67</b>
	10	9.52	<b>9.53</b>
	15	<b>7.06</b>	6.98
U	5	5.06	<b>5.29</b>
	10	7.93	<b>8.38</b>
	15	5.12	<b>5.17</b>

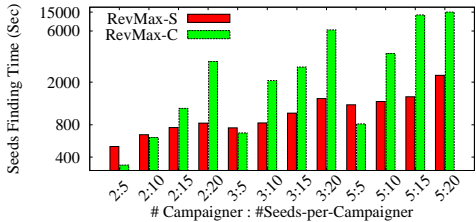


Fig. 9: Seed Sets Finding Time, K-LT Influence Cascading Model, *NetHEPT* Dataset

2) *Performance with K-LT Model:* We first illustrate in Tables IV and V the performance over the *NetHEPT* dataset by varying the number of campaigners from 2 to 5, number of seed nodes per campaigner from 5 to 15, and with three different revenue distributions: uniform, clustering with low competition, and clustering with high competition. We observe the following for the K-LT model. With more campaigners as well with more seed nodes, our heuristic method RevMax-S often outperforms our approximated technique RevMax-C. This is because the performance guarantee provided by RevMax-C is  $\frac{1}{m}(1 - \frac{1}{e})$ , which decreases with  $m$ , i.e., the number of campaigners.

We show the efficiency results over the *NetHept* dataset and with the K-LT model in Figure 9. We find that up to 5 seed nodes per campaigner, along with 2, 3, or even 5 campaigners, RevMax-C requires smaller running time as compared to that of RevMax-S. This is due to how the underlying viral marketing algorithm (i.e., CELF++ [11]) is applied differently in both these methods. For RevMax-C, CELF++ is applied only once to identify all the  $mk$  seed nodes; while for RevMax-S, CELF++ is applied  $m$  times — each time it identifies  $k$  seed nodes for one campaigner. However, CELF++ itself is an iterative algorithm; more specifically, it requires  $k$  iterations to identify the top- $k$  seed nodes. In one run of the CELF++ algorithm, the first iteration is the most expensive, and the subsequent iterations are significantly faster. In RevMax-S, the first iteration of CELF++ runs for  $m$  times;

TABLE VI: Revenue Improvement Rate (RIR), K-LT Influence Cascading Model, CRL Revenue Distribution, *DBLP* Dataset

#Seed Nodes per Camp.	# Camp.=2		# Camp.=3	
	RIR RevMax-S	RIR RevMax-C	RIR RevMax-S	RIR RevMax-C
5	77.27	<b>77.73</b>	27.41	<b>27.42</b>
10	42.27	<b>46.78</b>	40.80	<b>41.28</b>
15	<b>39.19</b>	37.04	36.74	<b>37.05</b>
20	<b>42.03</b>	42.01	<b>40.14</b>	34.25
25	31.01	<b>31.62</b>	<b>39.78</b>	34.90

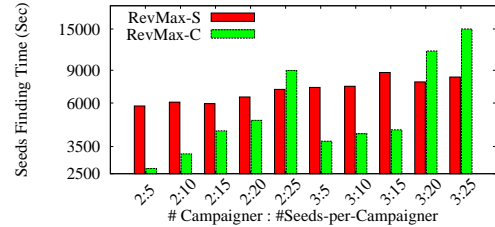


Fig. 10: Seed Sets Finding Time, K-LT Model, *DBLP* Dataset

while in RevMax-C, the first iteration of CELF++ runs only once. This explains why our proposed method RevMax-C is faster compared to the RevMax-S. Nevertheless, RevMax-C requires more time as one increases the number of seed nodes per campaigner. This is because the pruning technique in the CELF++ algorithm starts deteriorating with increasing number of seed nodes. Since RevMax-C directly identifies  $mk$  seed nodes, whereas RevMax-S iterates for  $k$  times and in each iteration, it identifies  $m$  seed nodes; RevMax-C takes more time for higher values of  $m$  or  $k$ .

We show the performance of RevMax-C and RevMax-S under the K-LT model over *DBLP* and *Flickr* in Tables VI and VII, respectively. We find very similar characteristics as before. When the campaigners are constrained by a small number of seed nodes, RevMax-C almost always outperforms RevMax-S both in terms of revenue improvement rate as well as in terms of the running time to identify the seed sets. However, RevMax-S starts performing well with more campaigners and more seed nodes.

3) *Uneven Revenue and Unequal Seed Set Sizes:* In these experiments, we consider two uneven revenue distributions, namely NE (Table VIII) and CNC (Table IX); as well as the scenario when the campaigners allow for different number of seed nodes (Table X). While our approximated technique RevMax-C outperforms our heuristic method RevMax-S, one may found that RevMax-S still very competitive with respect to RevMax-C.

### C. Scalability

In Figure 12, we analyze the variation of running times of RevMax-C and RevMax-S with different graph sizes. In particular, we consider varying sizes of the *Flickr* dataset, while keeping the number of campaigners and number of seeds per campaigner fixed at 2 and 5, respectively. We find that the running time of RevMax-C increases log-linearly with increasing graph sizes under the MCIC model, while it increases almost linearly with increasing graph sizes under the K-LT model.

In Figure 13, we illustrate the scalability of our heuristic method RevMax-S with the number of seed nodes per campaigner (up to 100) and also with the number of campaigners

TABLE VII: Revenue Improvement Rate (RIR), K-LT Influence Cascading Model, 5 Seeds/Campaigner, *Flickr* Dataset

Revenue Distribution	# Camp.=2		# Camp.=3	
	RIR RevMax-S	RIR RevMax-C	RIR RevMax-S	RIR RevMax-C
CRH	6.08	<b>6.27</b>	6.67	<b>6.79</b>
CRL	6.00	<b>6.31</b>	<b>6.70</b>	6.59
U	<b>20.14</b>	19.99	<b>13.75</b>	13.21

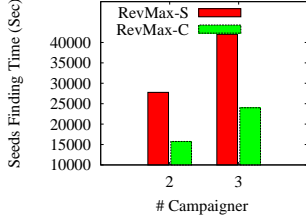


Fig. 11: Seed Sets Finding Time, K-LT Influence Cascading Model, 5 Seeds/Campaigners, *Flickr* Dataset

TABLE VIII: Revenue Improvement Rate (RIR) with Uneven NE Revenue Distribution, 2 Campaigners, *NetHEPT* Dataset

#Seed Nodes per Camp.	MCIC		K-LT	
	RIR RevMax-S	RIR RevMax-C	RIR RevMax-S	RIR RevMax-C
5	6.92	<b>8.99</b>	5.28	<b>5.55</b>
10	3.01	<b>3.48</b>	7.50	<b>7.63</b>
15	1.73	<b>1.85</b>	5.26	<b>5.53</b>
20	2.37	<b>2.51</b>	5.80	<b>5.89</b>

TABLE IX: Revenue Improvement Rate (RIR) with Uneven CNC Revenue Distribution, 2 Campaigners, *NetHEPT* Dataset

#Seed Nodes per Camp.	MCIC		K-LT	
	RIR RevMax-S	RIR RevMax-C	RIR RevMax-S	RIR RevMax-C
5	3.58	<b>3.83</b>	<b>8.19</b>	8.13
10	2.67	<b>2.93</b>	9.76	<b>9.80</b>
15	1.78	<b>2.22</b>	7.45	<b>7.54</b>
20	2.35	<b>2.64</b>	6.91	<b>6.98</b>

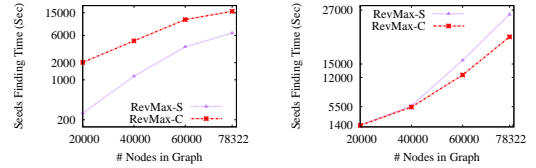
TABLE X: Revenue Improvement Rate (RIR) with Unequal Seed Set Sizes, CHC Revenue, 2 Campaigners, *NetHEPT*

#Seed Nodes $C_1$   $C_2$	MCIC		K-LT	
	RIR RevMax-S	RIR RevMax-C	RIR RevMax-S	RIR RevMax-C
5   10	2.30	<b>2.58</b>	5.13	<b>5.27</b>
10   20	1.74	<b>1.90</b>	5.23	<b>5.54</b>
15   30	1.42	<b>1.60</b>	5.26	<b>5.39</b>
20   40	1.48	<b>1.70</b>	4.99	<b>5.10</b>

(up to 10). Our results show that RevMax-S scales linearly with the number of campaigners and also with the number of seed nodes per campaigner.

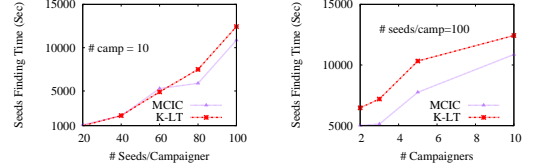
#### D. Summary

We summarize our experimental results below: (1) Our approximated techniques RevMax-C often outperforms our heuristic method RevMax-S, usually by a margin of 5%~10%. Nevertheless, RevMax-S is very competitive with respect to RevMax-C. (2) Both RevMax-S and RevMax-C are scalable with respect to the input graph size. (3) While RevMax-C suffers from scalability issues with increasing number of seed nodes and with many campaigners, RevMax-S is scalable even with a large number of campaigners and many seed nodes.



(a) MCIC Model (b) K-LT Model

Fig. 12: Scalability: Seed Sets Finding Time vs. Varying Graph Sizes, *Flickr* Dataset, 2 Campaigners with 5 Seeds/Campaigner



(a) Varying #Seeds (b) Varying #Campaigners

Fig. 13: Scalability of RevMax-S: Seed Sets Finding Time vs. Varying Seed Set Size and Campaigners, *NetHEPT* Dataset

## VIII. CONCLUSIONS

In this paper, we formulate and investigate the novel problem of revenue maximization of a social network host that sells viral marketing campaigns to multiple client campaigners. While our problem under both IC and LT models of influence cascading is NP-hard, and neither monotonic, nor sub-modular; we develop effective algorithms with theoretical performance guarantees. In addition, our proposed techniques can solve the revenue maximization problem exactly in polynomial time over a tree dataset using both these models. For scalability reasons, we also design efficient heuristics. Our experimental evaluation conducted on various real-world graph datasets and with diverse settings of revenue distributions attest high-quality and efficiency of our proposed techniques.

## REFERENCES

- [1] J. L. Aaker, A. M. Brumbaugh, and S. A. Grier. Nontarget Markets and Viewer Distinctiveness: The Impact of Target Marketing on Advertising Attitudes. *Consumer Psychology*, 9(3):127–140, 2000.
- [2] S. Bharathi, D. Kempe, and M. Salek. Competitive Influence Maximization in Social Networks. In *WINE*, 2007.
- [3] A. Borodin, Y. Filmus, and J. Oren. Threshold Models for Competitive Influence in Social Networks. In *WINE*, 2010.
- [4] C. Budak, D. Agrawal, and A. E. Abbadi. Limiting the Spread of Misinformation in Social Networks. In *WWW*, 2011.
- [5] T. Carnes, C. Nagarajan, S. M. Wild, and A. v. Zuylen. Maximizing Influence in a Competitive Social Network: A Follower’s Perspective. In *ICEC*, 2007.
- [6] W. Chen, A. Colin, R. Cumming, T. Ke, Z. Liu, D. Rincon, X. Sun, Y. Wang, W. Wei, and Y. Yuan. Influence Maximization in Social Networks when Negative Opinions May Emerge and Propagate. In *SDM*, 2011.
- [7] W. Chen, C. Wang, and Y. Wang. Scalable Influence Maximization for Prevalent Viral Marketing in Large-Scale Social Networks. In *KDD*, 2010.
- [8] W. Chen, Y. Wang, and S. Yang. Efficient Influence Maximization in Social Networks. In *KDD*, 2009.
- [9] P. Domingos and M. Richardson. Mining the Network Value Customers. In *KDD*, 2001.
- [10] H. N. Gabow, Z. Galil, T. Spencer, and R. E. Tarjan. Efficient Algorithms for Finding Minimum Spanning Trees in Undirected and Directed Graphs. *Combinatorica*, 6(2), 1986.



- [11] A. Goyal, W. Lu, and L. V. S. Lakshmanan. CELF++: Optimizing the Greedy Algorithm for Influence Maximization in Social Networks. In *WWW*, 2011.
- [12] S. Goyal and M. Kearns. Competitive Contagion in Networks. In *STOC*, 2012.
- [13] D. Kempe, J. Kleinberg, and E. Tardos. Maximizing the Spread of Influence through Social Network. In *KDD*, 2003.
- [14] M. Kimura and K. Saito. Tractable Models for Information Diffusion in Social Networks. In *PKDD*, 2006.
- [15] T. Lappas, E. Terzi, D. Gunopulos, and H. Mannila. Finding Effectors in Social Networks. In *KDD*, 2010.
- [16] J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. VanBriesen, and N. Glance. Cost-effective Outbreak Detection in Networks. In *KDD*, 2007.
- [17] F.-H. Li, C.-T. Li, and M.-K. Shan. Labeled Influence Maximization in Social Networks for Target Marketing. In *SocialCom*, 2011.
- [18] H. Li, S. S. Bhowmick, J. Cui, Y. Gao, and J. Ma. GETREAL: Towards Realistic Selection of Influence Maximization Strategies in Competitive Networks. In *SIGMOD*, 2015.
- [19] W. Lu, F. Bonchi, A. Goyal, and L. V. S. Lakshmanan. The Bang for the Buck: Fair Competitive Viral Marketing from the Host Perspective. In *KDD*, 2013.
- [20] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher. An Analysis of Approximations for Maximizing Submodular Set Functions. *Mathematical Programming*, 14(1):265–294, 1978.
- [21] M. Potamias, F. Bonchi, A. Gionis, and G. Kollios. k-Nearest Neighbors in Uncertain Graphs. *PVLDB*, 2010.
- [22] Y. Tang, X. Xiao, and Y. Shi. Influence Maximization: Near-Optimal Time Complexity Meets Practical Efficiency. In *SIGMOD*, 2014.
- [23] V. Tzoumas, C. Amanatidis, and E. Markakis. A Game-Theoretic Analysis of a Competitive Diffusion Process over Social Networks. In *WINE*, 2012.

## APPENDIX

**Proof of Theorem 4:** Let us denote by  $S_{opt}$  the optimal seed sets for the revenue maximization problem (Problem 1) under the K-LT model. Then, the host’s maximum revenue can be written as:  $\sum_{u \in V} \sum_{i=1}^m [A_{i,u} \times Pr_{K-LT}(u, i, S_{opt})]$ . Here,  $Pr_{K-LT}(u, i, S_{opt})$  denotes the expected spread of campaign  $C_i$  from the seed sets  $S_{opt}$  to a node  $u$  following the K-LT model.

Next, let us denote by  $S^*$  the best seed sets for the optimistic seed selection problem (Problem 2). It is easy to verify that

$$\sum_{u \in V} \sum_{i=1}^m [A_{i,u} \times Pr_{K-LT}(u, i, S_{opt})] \leq \sum_{u \in V} [A_u \times Pr_{LT}(u, S^*)] \quad (13)$$

This is because  $A_u = \max_{i \in (1, m)} A_{i,u}$ . Hence, the left hand side must be at most  $\sum_{u \in V} [A_u \times \sum_{i=1}^m Pr_{K-LT}(u, i, S_{opt})]$ , which is equal to  $\sum_{u \in V} [A_u \times Pr_{LT}(u, S_{opt})]$ . The equality in the last step follows from the definition of the K-LT model, that is, the activation probability of a node under the K-LT model is independent of how the seed set  $S_{opt}$  is partitioned among multiple campaigners. Hence, if Inequality 13 were not true, then  $S^*$  is not the optimal solution to Problem 2, as clearly a better solution  $S_{opt}$  for Problem 2 exists, which is a contradiction.

Let us define  $V_i \subseteq V$  such that for each node  $u \in V_i$  :  $A_u = A_{i,u}$  and  $\forall i \neq j : V_i \cap V_j = \emptyset$  and  $\bigcup_{i=1}^m V_i = V$ .

The following holds, because the summation still goes over all nodes in  $V$ .

$$\begin{aligned} & \sum_{u \in V} [A_u \times Pr_{LT}(u, S^*)] \\ &= \sum_{i=1}^m \sum_{u \in V_i} [A_u \times Pr_{LT}(u, S^*)] \\ &= \sum_{i=1}^m \sum_{u \in V_i} [A_{i,u} \times Pr_{LT}(u, S^*)] \\ &\leq \sum_{i=1}^m \left( \sum_{u \in V} [A_{i,u} \times Pr_{LT}(u, S^*)] \right) \\ &= \sum_{i=1}^m \left( \sum_{w \in S^*} \mathcal{R}_i(w) \right) \end{aligned} \quad (14)$$

The equality in the last line follows from the definition of  $\mathcal{R}_i(w)$ , as the right side of the equality, i.e.,  $\sum_{w \in S^*} \mathcal{R}_i(w)$  means that we assign the whole seed set  $S^*$  to campaigner  $C_i$ .

Next, let us denote by  $\mathcal{R}_{K-LT}$  the host’s expected revenue corresponding to the optimal partition of  $S^*$ . Since we assume that each campaigner has the same number of seed nodes, one may verify that  $\mathcal{R}_{K-LT} \geq \frac{1}{m} \sum_{i=1}^m \sum_{w \in S^*} \mathcal{R}_i(w)$ , with the equality holds if and only if the individual revenue vectors are equal, that is,  $\forall w \in S^*$  and  $\forall i, j \in (1, m)$ , it holds  $\mathcal{R}_i(w) = \mathcal{R}_j(w)$ . Here,  $m$  is the number of campaigners.

By combining Inequalities 13 and 14, we get:

$$\mathcal{R}_{K-LT} \geq \frac{1}{m} \sum_{u \in V} \sum_{i=1}^m [A_{i,u} \times Pr_{K-LT}(u, i, S_{opt})] \quad (15)$$

Finally, considering the fact that the optimal selection problem (Problem 2) is NP-hard and our iterative hill-climbing method produces a solution which is at least  $(1 - \frac{1}{e})$  of the solution corresponding to the optimal seed set  $S^*$ , the overall approximation ratio of our method is given by  $\frac{1}{m} (1 - \frac{1}{e})$ . Hence, the theorem.

**Non-overlapping Seed Sets:** We recall that in our model of viral marketing, each node can be activated only once and by one of the many competitive campaigns; and also the node stays activated with that campaign until the end. This reflects the real-world scenario that due to various product adoption costs, an average user often adopts only one of the multiple competing products [18], [19]. Since the seed nodes for a campaigner are the early adopters of the corresponding campaign, one node cannot act as the seed node for multiple competing campaigners. Thus, although some node might be one of the top- $k$  seed nodes for more than one campaigners individually, we allow the seed sets to be non-overlapping in case there are multiple competitive campaigns running simultaneously in the network [18], [19]. We illustrate this scenario with an example below.

*Example 6:* In Figure 14, we assume that nodes  $v_1$  and  $v_2$  belong to the target set of campaigners  $C_1$ ; and  $V_4$  and  $V_5$  in the target set of campaigner  $C_2$ . We also assume that

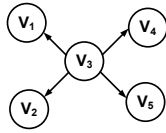


Fig. 14: Non-overlapping Seed Sets for Multiple Competitive Campaigns

TABLE XI: Revenue Improvement Rate (RIR), Uniform Revenue Distribution, *NetHEPT* Dataset, One Campaigner

#Seed Nodes	MCIC Model		K-LT Model	
	RIR RevMax-S	RIR RevMax-C	RIR RevMax-S	RIR RevMax-C
5	2.13	2.13	4.84	4.84
10	2.79	2.48	5.24	5.24
15	1.85	1.69	5.29	5.29
20	2.08	1.99	5.33	5.33

each campaigner has a budget of seed set size 1, while the probabilities on the edges are all 1. By following the IC model, one may note that  $V_3$  is the best seed node for both campaigners individually. However, in case of simultaneous campaigning, if  $V_3$  is already assigned to  $C_1$ , it would be more beneficial from  $C_2$ 's perspective if either of  $V_4$  or  $V_5$  is assigned as the seed node for her campaign.

Here, we must mention that due to the non-monotonicity property of the host's influence maximization problem (see Figure 4), it is not always advantageous to add a new client campaigner from the host's point of view, e.g., consider an extreme case when one campaigner provides very high revenues for all users in the network, and the other campaigner provides very low revenues for all users in the network. Nevertheless, in this work, we study the host's revenue maximization problem under the constraint that if the host has multiple client campaigners, she needs to serve all her client campaigners simultaneously by allocating them the specified number of distinct seed nodes. In real-world setting, this would be beneficial to the host for maintaining future relationships with all her clients.

**Experiments with  $m=1$  Campaigner:** In this section, we provide additional experimental results with  $m = 1$  campaigners. Note that if there is only one campaigner, the problem can be solved with the traditional viral marketing solution [11]. Thus, our objective here is to compare our proposed solutions with respect to the original methods in the presence of only one campaigner.

We note that our RevMax-S technique becomes identical to the traditional viral marketing solution with  $m = 1$  campaigner for both the MCIC and K-LT models. Now, our RevMax-C approach also becomes identical to the traditional viral marketing solution for the K-LT model. Therefore, our techniques for the K-LT model converges to the classical solution of influence maximization in the presence of  $m = 1$  campaigner (see Table XI).

For the MCIC model, one may recall that our RevMax-C solution is based on approximating the network by its most influential tree (Section IV). Therefore, with  $m = 1$  campaigner, we find that the traditional approach outperforms the RevMax-C technique often by a slight margin (see in Table XI that RevMax-S is better than RevMax-C often by a

TABLE XII: Tree Dataset Characteristics

Dataset	# Nodes	# Edges	Edge Prob: Mean, SD, Quartiles
<i>NetHEPT-Tree</i>	15 229	13 452	$0.25 \pm 0.28, \{0.0005, 0.21, 0.51\}$

TABLE XIII: Revenue Improvement Rate (RIR), MCIC Information Diffusion Model with 2 Campaigners, *NetHEPT-Tree* Dataset

Revenue Distribution	#Seed Nodes per Camp.	RIR	RIR
		RevMax-S	RevMax-C
CRL	5	15.89	16.51
	10	9.65	10.11
	15	8.65	9.37
	20	6.08	6.48
U	5	9.48	10.12
	10	10.47	11.09
	15	11.09	11.89
	20	9.46	10.06

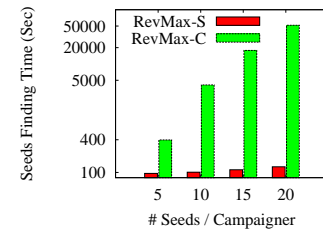


Fig. 15: Seed Sets Finding Time, MCIC Information Diffusion Model with 2 Campaigners, *NetHEPT-Tree* Dataset

small margin for the MCIC model). Thus, our results illustrate that while we certainly lose information by approximating a network with its most influential tree, our techniques are still very competitive with respect to the classical viral marketing solution in the presence of  $m = 1$  campaigner.

### Experiments with Tree Dataset:

**NetHEPT-Tree Dataset.** In order to verify the performance of our methods over a tree dataset, we also consider a spanning-tree (Table XII) of the original *NetHEPT* graph. The spanning tree is built by first randomly selecting a node and then identifying its breadth-first-search tree.

We now illustrate the performance of our techniques over tree datasets, since our algorithms can maximize the host's revenue over a tree dataset in an *exact* manner. We find in Table XIII that our method RevMax-C always outperforms the heuristic RevMax-S in terms of the host's revenue improvement rate. We also find very similar revenue improvement results over spanning trees of the two other datasets, i.e., *Flickr* and *DBLP*. Finally, figure 15 shows the corresponding seeds finding times over the *NetHEPT-Tree* dataset.