Boosting Information Spread: An Algorithmic Approach

Yishi Lin
Dept. of Computer Science and Engineering
The Chinese University of Hong Kong
yslin@cse.cuhk.edu.hk

Wei Chen
Microsoft Research
weic@microsoft.com

John C.S. Lui
Dept. of Computer Science and Engineering
The Chinese University of Hong Kong
cslui@cse.cuhk.edu.hk

***Abstract*—The majority of influence maximization (IM) studies focus on targeting influential *seeders* to trigger substantial information spread in social networks. In this paper, we consider a *new and complementary* problem of how to further increase the influence spread of given seeders. Our study is motivated by the observation that direct incentives could "boost" users so that they are more likely to be influenced by friends. We study the k-boosting problem which aims to find k users to boost so that the final "boosted" influence spread is maximized. The k-boosting problem is *different* from the IM problem because boosted users behave differently from seeders: *boosted* users are initially uninfluenced and we only increase their probability to be influenced. Our work also *complements* the IM studies because we focus on triggering larger influence spread on the basis of given seeders. Both the NP-hardness of the problem and the non-submodularity of the objective function pose challenges to the k-boosting problem. To tackle the problem, we devise two efficient algorithms with the *data-dependent* approximation ratio. We conduct extensive experiments using real social networks demonstrating the efficiency and effectiveness of our proposed algorithms. We show that boosting solutions returned by our algorithms achieves boosts of influence that are up to several times higher than those achieved by boosting solutions returned by intuitive baselines, which have no guarantee of solution quality. We also explore the "budget allocation" problem in our experiments. Compared with targeting seeders with all budget, larger influence spread is achieved when we allocation the budget to both seeders and boosted users. This also shows that our study complements the IM studies.**

I. Introduction

With the popularity of online social networks, *viral marketing*, which is a marketing strategy to exploit online *word-of-mouth* effects, has become a powerful tool for companies to promote sales. In viral marketing campaigns, companies target influential users by offering free products or services with the hope of triggering a chain reaction of product adoption. *Initial adopters* or *seeds* are often used interchangeably to refer to these targeted users. Motivated by the need for effective viral marketing strategies, *influence maximization* has become a fundamental research problem in the past decade. Most existing work on the influence maximization problems focused on the selection of initial adopters. In these studies, the goal is usually to maximize one's influence spread [1–8], or to block the influence spread of one's competitors [9, 10].

In practical marketing campaigns, companies often need to consider *a mixture of multiple promotion strategies* to make the best use of the marketing budget. Targeting influential users as initial adopters is one tactic, and we list some others as follows.

- *Customer incentive programs*: Companies offer incentives such as coupons, premiums, or product trials to attract potential customers. Targeted customers are in general more likely to be influenced or convinced by their friends.
- *Social media advertising*: Companies could reach intended audiences via digital advertising. According to the Nielsen Global Trust in advertising survey [11], owned online channels such as brand-managed sites are the second most trusted advertising formats (completely or somewhat trusted by 70% of global respondents), second only to recommendations from friends and family. We believe that customers who have been targeted by these advertisements are more likely to follow their friends' purchases.
- *Referral marketing*: Companies encourage customers to refer others to use the product by offering rewards such as cash back, special discounts, or gifts. In this case, targeted customers are more likely to influence their friends.

As one can see, these marketing strategies are able to "*boost*" the influence transferring through customers. Furthermore, for companies, the cost of "boosting" a customer (e.g., the average redemption and distribution cost per coupon, or the advertising cost per customer) is much lower than the cost of nurturing an influential user as an initial adopter and a product evangelist. Although identifying influential initial adopters have been actively studied, very little attention has been devoted to studying how to utilize incentive programs or other strategies to further increase the influence spread of initial adopters to a new level.

In this paper, we study the algorithmic problem of finding k boosted users so that when their friends adopt a product, they are more likely to make the purchase and continue to influence others. Motivated by the need for modeling boosted customers, we propose a novel *influence boosting model*, extending the *Independent Cascade* (IC) model. In the *influence boosting model*, *seed users* generate influence same as in the IC model. In addition, we introduce the *boosted user* as a new type of special users. Boosted users represent customers with incentives such as coupons. They are uninfluenced at the beginning of the influence propagation process. However, they are more likely to be influenced by their friends and further spread the influence to others. In other words, they are able to "boost" the influence transferring through them. Under the influence boosting model, we study *how to boost the influence spread with initial adopters given*. More precisely, given a set of initial adopters, we are interested in identifying k users among other users, so that the expected influence spread upon "boosting"

the identified users is maximized. Because of the essential differences in behaviors between seed users and boosted users, our work is very different from the influence maximization studies focusing on selecting seeds.

Our work also complements the studies of influence maximization problems. First, compared with nurturing an initial adopter, *boosting* a potential customer usually incurs a lower cost. For example, companies may need to offer free products to initial adopters, but only need to offer incentives such as discount coupons to boost potential customers. With both our methods that identify users to *boost* and influence maximization algorithms that select initial adopters, companies have more flexibility in determining where to allocate their marketing budgets. Second, initial adopters are sometimes predetermined. For example, the initial adopters and evangelist of a product may be advocates of a particular brand or prominent bloggers in the area, In this case, our "boosting" methodology suggests how to effectively utilize incentive programs or similar marketing strategies to take the influence spread to the next level.

Contributions. We study a novel problem of how to *boost* the influence spread when the initial adopters are given. We summarize our contributions as follows.

- We present the *influence boosting model*, which integrates the idea of *boosting* into the *Independent Cascade* model.
- We formulate a k-boosting problem that asks how to maximize the boost of influence spread under the *influence boosting model*. The k-boosting problem is NP-hard. Computing the *expected boost of influence spread* is #P-hard. Moreover, the boost of influence spread does not possess the submodularity, meaning that the greedy algorithm may not have any performance guarantee.
- We present efficient approximation algorithms `PRR-Boost` and `PRR-Boost-LB` for the k-boosting problem.
- We conduct extensive experiments using real networks with learned influence probabilities among users. Experimental results show the efficiency and effectiveness of our proposed algorithms, and demonstrate the superiority of our algorithms over intuitive baselines.

Paper organization. Section II provides background and related works. We describe the *influence boosting model* and the k-boosting problem in Section III. We present building blocks of `PRR-Boost` and `PRR-Boost-LB` for the k-boosting problem in Section IV, and the detailed algorithm design in Section V. We show experimental results in Section VI. Section VII concludes the paper.

II. Background and Related Work

In this section, we provide background information about influence maximization problems and review related works.

Classical influence maximization problems. Kempe et al. [1] first formulated the *influence maximization* problem. They proposed the *Independent Cascade* (IC) model to describe the influence diffusion process. Under the IC model, given a graph $G = (V, E)$, influence probabilities on edges and a set $S \subseteq V$ of *seeds*, the influence propagates as follows. Initially, nodes in S are activated. Each newly activated node u influences its neighbor v with probability p_{uv}. The *expected influence spread* of S is the expected number of nodes activated at the end of the influence diffusion process. The influence maximization problem is to select a set S of k nodes so that the expected influence spread is maximized. Under the IC model, the influence maximization problem is NP-hard [1] and computing the expected influence spread for a given S is #P-hard [4]. A series of studies have been done to approximate the influence maximization problem under the IC model or other diffusion models [3, 4, 6–8, 12–15].

Boost the influence spread. To boost the influence spread, several works studied how to recommend friends or inject links into social networks [16–21]. Lu et al. [22] studied how to maximize the expected number of adoptions by targeting initial adopters of a complementing product. Chen et al. [23] considered the *amphibious influence maximization*. They studied how to select a subset of seed content providers and a subset of seed customers so that the expected number of influenced customers is maximized. Their model differs from ours in that they only consider influence originators selected from content providers, which are separated from the social network, and influence boost is only from content providers to consumers in the social network. Yang et al. [21] studied how to offer discounts under an assumption different from ours. They assumed that the probability of a customer being an initial adopter is a known function of the discounts offered to him. They studied how to offer discounts to customers with a limited budget so that the influence cascades triggered is maximized. Different from the above studies, we study how to *boost the spread of influence* when seeds are given. We assume that we can give incentives to some users (i.e., "boost" some users) so that they are more likely to be influenced by their friends, but they themselves would not become adopters without friend influence.

III. Model and Problem Definition

In this section, we first present the *influence boosting model* and define the k-boosting problem. Then, we highlight the challenges associated with solving the proposed problem.

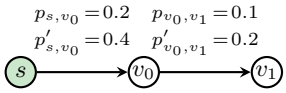
A. Model and Problem Definition

Traditional studies of the influence maximization problem focus on how to identify a set of k influential users (or *seeds*) who can trigger the largest influence diffusion. In this paper, we aim to *boost* the influence propagation assuming that *seeds* are given. We first define the *influence boosting model*.

Definition 1 (Influence Boosting Model)**.** *Suppose we are given a directed graph $G = (V, E)$ with n nodes and m edges, two influence probabilities p_{uv} and p'_{uv} (with $p'_{uv} > p_{uv}$) on each edge e_{uv}, a set $S \subseteq V$ of seeds, and a set $B \subseteq V$ of boosted nodes. The influence propagates in discrete time steps as follows. If v is not boosted, each of its newly-activated in-neighbor u influences v with probability p_{uv}. If v is a boosted node, each of its newly-activated in-neighbor u influences v with probability p'_{uv}.*

In Definition 1, we assume that "boosted" users are more likely to be influenced. It is important to note that our study can be easily adapt to the case where boosted users are more influential: if a newly-activated user u is boosted, she influences her neighbors with probability p'_{uv} instead of

p_{uv}. To simplify the presentation, we focus on the influence boosting model in Definition 1.



B	$\sigma_S(B)$	$\Delta_S(B)$
$\emptyset$	1.22	0.00
$\{v_0\}$	1.44	0.22
$\{v_1\}$	1.24	0.02
$\{v_0, v_1\}$	1.48	0.26

Fig. 1: Example of the influence boosting model ($S=\{s\}$).

Let $\sigma_S(B)$ be the expected influence spread of seeds in S upon boosting nodes in B. We refer to $\sigma_S(B)$ as the *boosted influence spread*. Let $\Delta_S(B) = \sigma_S(B) - \sigma_S(\emptyset)$. We refer to $\Delta_S(B)$ as the *boost of influence spread of* B, or simply the *boost of* B. To illustrate, consider the example in Figure 1. We have $\sigma_S(\emptyset) = 1.22$, which is essentially the influence spread of $S = \{s\}$ in the IC model. When we boost node v_0, we have $\sigma_S(\{v_0\}) = 1 + 0.4 + 0.04 = 1.44$, and $\Delta_S(\{v_0\}) = 1.44 - 1.22 = 0.22$. We now formulate the *k-boosting problem*.

Definition 2 (k-Boosting Problem)**.** *Given a directed graph $G = (V, E)$, influence probabilities p_{uv} and p'_{uv}'s on ever edges e_{uv}, and a set $S \subseteq V$ of seed nodes, find a boost set $B \subseteq V$ with k nodes, such that the boost of influence spread of B is maximized. That is, determine $B^* \subseteq V$ such that*

$$B^* = \arg\max_{B \subseteq V, |B| \leq k} \Delta_S(B). \quad (1)$$

Remarks. By definition, one can see that the k-boosting problem is very different from the classical influence maximization problem. In addition, we observe that boosting nodes that significantly increase the influence spread when used as additional seeds could be extremely inefficient. For example, consider the example in Figure 1. If we are allowed to select one more seed, node v_1 is the optimal choice. However, if we can boost a node, boosting v_0 is much better than boosting v_1. Section VI provides more experimental results.

B. Challenges of the Boosting Problem

In this part, we analyze several key properties of the k-boosting problem and show the challenges we face. Theorem 1 indicates the hardness of the k-boosting problem.

Theorem 1 (Hardness)**.** *The k-boosting problem is NP-hard. The computation of $\Delta_S(B)$ given S and B is #P-hard.*

Proof outline: The NP-hardness is proved by a reduction from the NP-complete *Set Cover* problem [24]. The #P-hardness of the computation is proved by a reduction from the #P-complete counting problem of *s-t connectedness* in directed graphs [25]. The full analysis can be found in our technical report [26]. ■

Non-submodularity of the boosted influence. Because of the above hardness results, we explore approximation algorithms to tackle the k-boosting problem. In most influence maximization problems, the expected influence of the seed set S (i.e., the objective function) is a monotone and submodular function of S.[1] Thus, a natural greedy algorithm returns a solution with an approximation guarantee [1, 6–8, 15, 27]. However, the objective function $\Delta_S(B)$ in our problem is *neither submodular nor supermodular* on the set B of boosted nodes. On one hand, when we boost several nodes on different parallel paths from seed nodes, their overall boosting effect exhibits a *submodular behavior*. On the other hand, when we boost several nodes on a path starting from a seed node, their boosting effects can be cumulated along the path, generating a larger overall effect than the sum of their individual boosting effect. This is in fact a *supermodular behavior*. To illustrate, consider the graph in Figure 1, we have $\Delta_S(\{v_0, v_1\}) - \Delta_S(\{v_0\}) = 0.04$, which is larger than $\Delta_S(\{v_1\}) - \Delta_S(\emptyset) = 0.02$. In general, the boosted influence has a complicated interaction between supermodular and submodular behaviors when the boost set grows, and is neither supermodular nor submodular. The non-submodularity of $\Delta_S(\cdot)$ indicates that the boosting set returned by the greedy algorithm may not have the $(1-1/e)$-approximation guarantee. Therefore, besides the NP-hardness of the problem and the #P-hardness of computing $\Delta_S(\cdot)$, the non-submodularity of the objective function poses an additional challenge.

[1] A set function f is monotone if $f(S) \leq f(T)$ for all $S \subseteq T$; it is submodular if $f(S \cup \{v\}) - f(S) \geq f(T \cup \{v\}) - f(T)$ for all $S \subseteq T$ and $v \notin T$, and it is supermodular if $-f$ is submodular.

IV. Boosting on General Graphs

In this section, we present three building blocks for solving the k-boosting problem: (1) a state-of-the-art influence maximization framework, (2) the *Potentially Reverse Reachable Graph* for estimating the boost of influence spread, and (3) the *Sandwich Approximation* strategy [22] for maximizing non-submodular functions. Our solutions to the k-boosting problem integrate the three building blocks. We will present the detailed algorithm design in the next section.

A. State-of-the-art influence maximization techniques

In influence maximization problems, we want to select k seeds so that the expected influence spread is maximized. One state-of-the-art method is the *Influence Maximization via Martingale* (`IMM`) method [8] based on the idea of *Reverse-Reachable Sets* (RR-sets) [6]. We use the `IMM` method in this work, but it is important to note that the `IMM` method could be replaced by other similar influence maximization frameworks based on RR-sets (e.g., `TIM/TIM+` [7] or `SSA/D-SSA` [15]).

RR-sets. An *RR-set* for a node r is a random set R of nodes, such that for any seed set $S \subseteq V$, the probability that $R \cap S \neq \emptyset$ equals the probability that r can be activated by S in a random diffusion process. Node r may also be selected uniformly at random from V, and the RR-set will be generated accordingly with the random node r. One key property of RR-sets is that the expected influence of S equals to $n \cdot \mathbb{E}[\mathbb{I}(R \cap S \neq \emptyset)]$ for all $S \subseteq V$, where $\mathbb{I}(\cdot)$ is the indicator function and the expectation is taken over the randomness of R.

General `IMM` algorithm. The `IMM` algorithm has two phases. The *sampling* phase keeps generating random RR-sets until a stopping criteria is met, indicating that the estimation of the influence spread is "accurate enough". The *node selection* phase greedily selects k seed nodes based on the generated RR-sets. Under a diffusion model where generating a random RR-set takes time $O(EPT)$, the `IMM` algorithm returns a $(1 - 1/e - \epsilon)$-approximate solution with at least $1 - n^{-\ell}$ probability, and runs in $O(\frac{EPT}{OPT} \cdot (k + \ell)(n + m) \log n / \epsilon^2)$ expected time, where OPT is the optimal expected influence.

B. Potentially Reverse Reachable Graphs

We now describe how we estimate the boost of influence, which is our objective function in the k-boosting problem. The estimation is based on the concept of the *Potentially Reverse Reachable Graph* (PRR-graph), which is defined as follows.

Definition 3 (Potentially Reverse Reachable Graph)**.** *Let r be a node in G. A Potentially Reverse Reachable Graph (PRR-graph) R for a node r is a random graph generated as follows. We first sample a deterministic copy g of G. In the deterministic graph g, each edge e_{uv} in graph G is "live" in g with probability p_{uv}, "live-upon-boost" with probability $p'_{uv} - p_{uv}$, and "blocked" with probability $1 - p'_{uv}$. The PRR-graph R is the minimum subgraph of g containing all paths from seed nodes to r through non-blocked edges in g. We refer to r as the "root node". When r is also selected from V uniformly at random, we simply refer to the generated PRR-graph as a random PRR-graph (for a random root).*

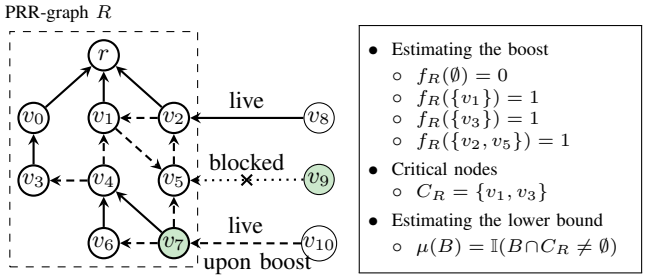


Fig. 2: Example of a Potentially Reverse Reachable Graph.

Figure 2 shows an example of a PRR-graph R. The directed graph G contains 12 nodes and 16 edges. Node r is the *root* node. Shaded nodes are seed nodes. Solid, dashed and dotted arrows with crosses represent live, live-upon-boost and blocked edges, respectively. The PRR-graph for r is the subgraph in the dashed box. Nodes and edges outside the dashed box do not belong to the PRR-graph. It is easy to check that nodes and edges outside the dashed box are not on any paths from seed nodes to r that only contain non-blocked edges. By definition, a PRR-graph may contain loops. For example, the PRR-graph in Figure 2 contains a loop among nodes v_1, v_5, and v_2.

Estimating the boost of influence. Let R be a given PRR-graph with root r. By definition, every edge in R is either *live* or *live-upon-boost*. Given a path in R, we say that it is *live* if and only if it contains only live edges. Given a path in R and a set of boosted nodes $B \subseteq V$, we say that the path is *live upon boosting* B if and only if the path is not a *live* one, but every edge e_{uv} on it is either *live* or *live-upon-boost* with $v \in B$. For example, in Figure 2, the path from v_3 to r is *live*, and the path from v_7 to r via v_4 and v_1 is *live upon boosting* $\{v_1\}$. Define $f_R(B) : 2^V \to \{0, 1\}$ as: $f_R(B) = 1$ if and only if, in R, (1) there is no *live* path from seed nodes to r; and (2) a path from a seed node to r is *live upon boosting* B. Intuitively, in the deterministic graph R, $f_R(B) = 1$ if and only if the root node is inactive without boosting, and active upon boosting nodes in B. In Figure 2, if $B = \emptyset$, there is no live path from the seed node v_7 to r upon boosting B. Therefore, we have $f_R(\emptyset) = 0$. There is a live path from the seed node v_7 to r that is live upon boosting $\{v_1\}$, and thus we have $f_R(\{v_1\}) = 1$. Similarly, we have $f_R(\{v_3\}) = f_R(\{v_2, v_5\}) = 1$. Based on the above definition of $f_R(\cdot)$, we have the following lemma.

Lemma 1. *For any $B \subseteq V$, we have $n \cdot \mathbb{E}[f_R(B)] = \Delta_S(B)$, where the expectation is taken over the randomness of R.*

Proof: For a random PRR-graph R whose root node is randomly selected, $\Pr[f_R(B) = 1]$ equals the difference between probabilities that a random node in G is activated given that we boost B and we do not boost. ■

Let $\mathcal{R}$ be a set of independent random PRR-graphs, define

$$\hat{\Delta}_{\mathcal{R}}(B) = \frac{n}{|\mathcal{R}|} \cdot \sum_{R \in \mathcal{R}} f_R(B), \forall B \subseteq V. \tag{2}$$

Based on the Chernoff bound, $\hat{\Delta}_{\mathcal{R}}(B)$ closely estimates $\Delta_S(B)$ for any $B \subseteq V$ if $|\mathcal{R}|$ is sufficiently large.

C. Sandwich Approximation Strategy

To tackle the non-submodularity of function $\Delta_S(\cdot)$, we apply the *Sandwich Approximation* (SA) strategy [22]. Using notations of our k-boosting problem, the SA strategy [22] works as follows. First, we find submodular lower and upper bound functions of Δ_S, denoted by μ and ν. Then, we select node sets B_Δ, B_μ and B_ν by greedily maximizing Δ_S, μ and ν under the cardinality constraint of k. Ideally, we return $B_{\text{sa}} = \arg\max_{B \in \{B_\mu, B_\nu, B_\Delta\}} \Delta_S(B)$ as the final solution. Let the optimal solution of the k-boosting problem be B^* and let $OPT = \Delta_S(B^*)$. Suppose B_μ and B_ν are $(1 - 1/e - \epsilon)$-approximate solutions for maximizing μ and ν, we have

$$\Delta_S(B_{\text{sa}}) \geq \frac{\mu(B^*)}{\Delta_S(B^*)} \cdot (1 - 1/e - \epsilon) \cdot OPT, \tag{3}$$

$$\Delta_S(B_{\text{sa}}) \geq \frac{\Delta_S(B_\nu)}{\nu(B_\nu)} \cdot (1 - 1/e - \epsilon) \cdot OPT. \tag{4}$$

Therefore, to obtain a good approximation guarantee, at least one of μ and ν should be close to Δ_S. In this work, we derive a submodular lower bound μ of Δ_S using the definition of PRR-graphs. Because μ is significantly closer to Δ_S than any submodular upper bound we have tested, we only use the lower bound function μ and the "lower-bound side" of the SA strategy with approximation guarantee shown in Ineq. (3).

Submodular lower bound. We now derive a submodular lower bound of Δ_S. Let R be a PRR-graph with the root node r. Let $C_R = \{v | f_R(\{v\}) = 1\}$. We refer to nodes in C_R as *critical nodes* of R. Intuitively, the root node r becomes activated if we boost any node in C_R. For any node set $B \subseteq V$, define $f_R^-(B) = \mathbb{I}(B \cap C_R \neq \emptyset)$. By definition of C_R and $f_R^-(\cdot)$, we have $f_R^-(B) \leq f_R(B)$ for all $B \subseteq V$. Moreover, because the value of $f_R^-(B)$ is based on whether the node set B intersects with a fixed set C_R, $f_R^-(B)$ is a submodular function on B. For any $B \subseteq V$, define $\mu(B) = n \cdot \mathbb{E}[f_R^-(B)]$ where the expectation is taken over the randomness of R. Lemma 2 shows the properties of the function μ.

Lemma 2. *We have $\mu(B) \leq \Delta_S(B)$ for all $B \subseteq V$. Moreover, $\mu(B)$ is a submodular function of B.*

Proof: For all $B \subseteq V$, we have $\mu(B) \leq \Delta_S(B)$ because $f_R^-(B) \leq f_R(B)$ holds for any PRR-graph R. Moreover, $\mu(B)$

is submodular on B because $f_R^-(B)$ is a submodular function of B for any PRR-graph R. ■

Our experiments show that μ is close to Δ_S, especially for small values of k, say less than a thousand. Define

$$\hat{\mu}_{\mathcal{R}}(B) = \frac{n}{|\mathcal{R}|} \cdot \sum_{R \in \mathcal{R}} f_R^-(B), \forall B \subseteq V.$$

Because $f_R^-(B)$ is submodular on B for any PRR-graph R, $\hat{\mu}_{\mathcal{R}}(B)$ is submodular on B. Moreover, by Chernoff bound, $\hat{\mu}_{\mathcal{R}}(B)$ is close to $\mu(B)$ when $|\mathcal{R}|$ is sufficiently large.

Remarks on function $\mu(B)$. Function $\mu(B)$ does correspond to some physical diffusion model. Roughly speaking, $\mu(B)$ is the influence spread in a diffusion model with the boost set B, and the constraint that at most one edge on the influence path from a seed node to an activated node can be boosted. Due to space limit, we omit the precise description of the corresponding diffusion model here and include it in [26]. Compared with the convoluted diffusion model corresponding to $\mu(B)$, the PRR-graph description of $\mu(B)$ is more direct and is easier to analyze. Our insight is that by fixing the randomness in the original diffusion model, it may be easier to derive submodular lower-bound or upper-bound functions.

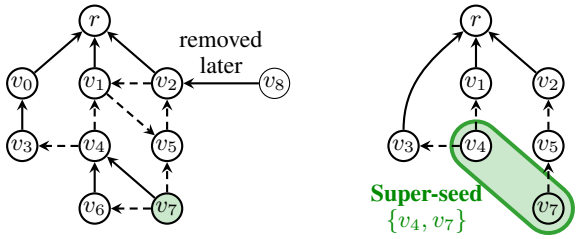
V. Algorithm Design

In this section, we first present how we generate random PRR-graphs. Then we obtain overall algorithms for the k-boosting problem by integrating the general `IMM` algorithm with PRR-graphs and the *Sandwich Approximation* strategy.

A. Generating PRR-graphs

We classify PRR-graphs into three categories. Let R be a PRR-graph with root node r. (1) *Activated*: If there is a *live* path from a seed node to r; (2) *Hopeless*: If there is no seeds in R, or there is no path from seeds to r with at most k *non-live* edges; (3) *"Boostable"*: not the above two categories. If R is not *boostable* (i.e. case (1) or (2)), we have $f_R(B) = f_R^-(B) = 0$ for all $B \subseteq V$. Therefore, for "non-boostable" PRR-graphs, we only count their occurrences and we terminate the generation of them once we know they are not *boostable*. Algorithm 1 depicts how we generate a random PRR-graph. The generation contains two phases. The first phase (Lines 1-19) generates a PRR-graph R. If R is boostable, the second phase compresses R to reduce its size. Figure 3 shows the results of two phases, given that the status sampled for every edge is same as that in Figure 2.

Phase I: Generating a PRR-graph. Let r be a random node. In this phase, a backward Breadth-First Search (BFS) from r makes sure that all *non-blocked* paths from seed nodes to r with at most k *live-upon-boost* edges are included in R. The status of each edge (i.e., *live, live-upon-boost, blocked*) is sampled when we first process it. The detailed backward BFS is as follows. Define the *distance* of a path from u to v as the number of *live-upon-boost* edges on it. Then, the *shortest distance* from v to r is the minimum number of nodes we have to boost so that at least a path from v to r becomes live. For example, in Figure 3a, the shortest distance from v_7 to r is one. During the generation of R, we use $d_r[\cdot]$ to maintain the shortest distances from nodes to the root node r. Initially,



(a) Results of phase I (b) Results of phase II

Fig. 3: Generation of a PRR-Graph. (Solid and dashed arrows represent live and live-upon-boost edges respectively.)

Algorithm 1: Generating a random PRR-graph (G, S, k)

```
1  Select a random node r as the root node
2  if r ∈ S then return R is activated
3  Create a graph R with a singleton node r
4  Create a double-ended queue Q with (r, 0)
5  Initialize d_r[r] ← 0 and d_r[v] ← +∞, ∀v ≠ r
6  while Q is not empty do
7      (u, d_ur) ← Q.dequeue_front()
8      if d_ur > d_r[u] then continue      // we've processed u
9      for each non-blocked incoming edge e_vu of u do
10         d_vr ← 𝕀(e_vu is live-upon-boost) + d_ur
11         if d_vr > k then continue                // pruning
12         Add e_vu to R
13         if d_vr < d_r[v] then
14             d_r[v] ← d_vr
15             if v ∈ S then
16                 if d_r[v] = 0 then return R is activated
17             else if d_vr = d_ur then Q.enqueue_front((v, d_vr))
18             else Q.enqueue_back((v, d_vr))
19 if there is no seed in R then return R is hopeless
20 Compress the boostable R to reduce its size
21 return a compressed boostable R
```

we have $d_r[r] = 0$ and we enqueue $(r, 0)$ into a double-ended queue Q. We repeatedly dequeue and process a node-distance pair (u, d_{ur}) from the head of Q, until the queue is empty. Note that the distance d_{ur} in a pair (u, d_{ur}) is the shortest known distance from u to r when the pair was enqueued. Thus we may find $d_{ur} > d_r[u]$ in Line 8. Pairs (u, d_{ur}) in Q are in the ascending order of the distance d_{ur} and there are at most two different values of distance in Q. Therefore, we process nodes in the ascending order of their shortest distances to r. When we process a node u, for each of its non-blocked incoming edge e_{vu}, we let d_{vr} be the shortest distance from v to r via u. If $d_{vr} > k$, all paths from v to r via u are impossible to become live upon boosting at most k nodes, therefore we ignore e_{vu} safely in Line 11. This is in fact a "pruning" strategy, because it may reduce unnecessary costs in the generation step. The pruning strategy is effective for small values of k. For large values of k, only a small number of paths need to be pruned due to the small-world property of real social networks. If $d_{vr} \leq k$, we insert e_{vu} into R, update $d_r[v]$ and enqueue (v, d_{vr}) if necessary. During the generation, if we visit a seed node s and its shortest distance to r is zero, we know R is

activated and we terminate the generation (Line 16). If we do not visit any seed node during the backward BFS, R *is hopeless* and we terminate the generation (Line 19).

Remarks on Phase I. At the end of phase I, R may include nodes and edges not belonging to it (e.g., *non-blocked* edges not on any non-blocked paths from seeds to the root). These *extra* nodes and edges will be removed in the compression phase. For example, Figure 3a shows the results of the first phase, given that we are constructing a PRR-graph R according to the root node r and sampled edge status shown in Figure 2. At the end of the first phase, we also include the *extra* edge from v_8 to v_2 in R, and they will be removed in the next phase.

Phase II: Compressing the PRR-graph. When we reach Line 20, R is *boostable*. In practice, we observe that we can remove and merge a significant fraction of nodes and edges from R (i.e., *compress* R), while keeping values of $f_R(B)$ and $f_R^-(B)$ for all $|B| \leq k$ same as before. Therefore, we compress all boostable PRR-graphs to prevent the memory usage from becoming a bottleneck. Figure 3b shows the compressed result of Figure 3a. *First*, we merge nodes v_4 and v_7 into a single "super-seed" node, because they are activated without boosting any node. Then, we remove node v_6 and its incident edges, because they are not on any paths from the super-seed node to the root node r. Similarly, we remove the extra node v_8 and the extra edge from v_8 to v_2. *Next*, observing that there are live paths from nodes v_0, v_1, v_2 and v_3 to root r, we remove all outgoing edges of them, and add a direct live edge from each of these nodes to r. After doing so, we remove node v_0 because it is not on any path from the super-seed node to r. Now, we describe the compression phase in detail.

The *first part* of the compression merges nodes into the super-seed node. We run a forward BFS from seeds in R to the root node r. For every node v, we compute the shortest distance $d_S[v]$ from seeds to v. If $d_S[v] = 0$ for a node v, there is a live path from a seed node to v in R. Let $X = \{v | d_S[v] = 0\}$, whether we boost any subset of X has nothing to do with whether the root node of R is activated. Thus, we have $f_R(B) = f_R(B \backslash X)$ for all $B \subseteq V$. To compress R, we merge all nodes in X as a single super-seed node: we insert a super-seed node x into R; for every node v in X, we remove its incoming edges and redirect every of its outgoing edge e_{vu} to e_{xu}. Finally, we clean up nodes and edges not on any paths from the super-seed node to the root node r.

In the *second part*, we add live edges so that nodes connected to r through live paths are directly linked to r. We also clean up nodes and edges that are not necessary for later computation. For a node v, let $d'_r[v]$ be the shortest distance from v to r without going through the super-seed. If a node v satisfies $d'_r[v] + d_S[v] > k$, every path going through v cannot be live with at most k nodes boosted, therefore we remove v and its adjacent edges. If a non-root node v satisfies $d'_r[v] = 0$, we remove its outgoing edges and add a live edge from v to r. In fact, in a boostable R, if a node v satisfies $d'_r[v] = 0$, we must have $d_r[v] = 0$ in the first phase. In our implementation, if a node v satisfies $d_r[v] = 0$, we in fact clear outgoing edges of v and add the live edge e_{vr} to R in the first phase. Finally, we remove "non-super-seed" nodes with no incoming edges.

Time complexity. The cost for the first phase of the PRR-graph generation is linear to the number of edges visited during the generation. The compression phase runs linear to the number of uncompressed edges generated in the generation phase. Section VI shows the average number of uncompressed edges in boostable PRR-graphs in several social networks.

B. PRR-Boost Algorithm

We obtain our algorithm, `PRR-Boost`, by integrating PRR-graphs, the `IMM` algorithm and the *Sandwich Approximation* strategy. Algorithm 2 depicts `PRR-Boost`.

Algorithm 2: PRR-Boost$(G, S, k, \epsilon, \ell)$

1 $\ell' = \ell \cdot (1 + \log 3 / \log n)$
2 $\mathcal{R} \leftarrow$ `SamplingLB`$(G, S, k, \epsilon, \ell')$ // sampling in general `IMM` [8], using the PRR-graph generation of Algo. 1
3 $B_\mu \leftarrow$ `NodeSelectionLB`$(\mathcal{R}, k)$ // maximize μ
4 $B_\Delta \leftarrow$ `NodeSelection`$(\mathcal{R}, k)$ // maximize Δ_S
5 $B_{\text{sa}} = \arg\max_{B \in \{B_\Delta, B_\mu\}} \hat{\Delta}_{\mathcal{R}}(B)$
6 **return** B_{sa}

Lines 1-3 utilize the `IMM` algorithm [8] with the PRR-graph generation given in Algorithm 1 to maximize the lower bound μ of Δ_S under the cardinality constraint of k. Line 4 greedily selects a set B_Δ of nodes with the goal of maximizing Δ_S, and we reuse PRR-graphs in $\mathcal{R}$ to estimate $\Delta_S(\cdot)$. Finally, between B_μ and B_Δ, we return the set with a larger estimated boost of influence as the final solution.

Approximate ratio. Let B^*_μ be the optimal solution for maximizing μ under the cardinality constraint of k, and let $OPT_\mu = \mu(B^*_\mu)$. By the analysis of the `IMM` method, we have the following lemma about Algorithm 2.

Lemma 3. *Define* $\epsilon_1 = \frac{\epsilon \cdot \alpha}{(1-1/e)\alpha + \beta}$, *where* $\alpha = \sqrt{\ell' \log n + \log 2}$, *and* $\beta = \sqrt{(1-1/e) \cdot (\log \binom{n}{k} + \ell' \log n + \log 2)}$. *With a probability of at least* $1 - n^{-\ell'}$, *the number of PRR-graphs generated in Line 2 satisfies*

$$|\mathcal{R}| \geq \frac{(2-2/e) \cdot n \cdot \log\left(\binom{n}{k} \cdot 2n^{\ell'}\right)}{(\epsilon - (1-1/e) \cdot \epsilon_1)^2 \cdot OPT_\mu} \quad \textit{(Th.2 in [8])}. \quad (5)$$

Given that Ineq. (5) *holds, with a probability of at least* $1 - n^{-\ell'}$, *the set* B_μ *returned by Line 3 satisfies*

$$n \cdot \hat{\mu}_{\mathcal{R}}(B_\mu) \geq (1-1/e)(1-\epsilon_1) \cdot OPT_\mu \quad \textit{(Th.1 in [8])}. \quad (6)$$

Ideally, while applying the *Sandwich Approximation* strategy, we should select $B_{\text{sa}} = \arg\max_{B \in \{B_\mu, B_\Delta\}} \Delta_S(B)$. However, Theorem 1 have stated the #P-hardness of computing $\Delta_S(B)$ for a given B. In Line 5, we select B_{sa} from B_μ and B_Δ with the larger *estimated* boost of influence. The following lemma shows that boosting B_{sa} leads to a large expected boost.

Lemma 4. *Given that Ineq.* (5)-(6) *hold, with a probability of at least* $1 - n^{-\ell'}$, *we have*

$$\Delta_S(B_{sa}) \geq (1 - 1/e - \epsilon) \cdot OPT_\mu \geq (1 - 1/e - \epsilon) \cdot \mu(B^*).$$

Proof outline: Let B be a boost set with k nodes, we say that B is a *bad* set if $\Delta_S(B) < (1 - 1/e - \epsilon) \cdot OPT_\mu$. Let B be an arbitrary *bad* set with k nodes. If we return B, we must have $\hat{\Delta}_{\mathcal{R}}(B) > \hat{\Delta}_{\mathcal{R}}(B_\mu)$. Therefore, the probability that

B is returned is upper bounded by $\Pr[\hat{\Delta}_{\mathcal{R}}(B) > \hat{\Delta}_{\mathcal{R}}(B_\mu)]$. Moreover, we can prove that $\Pr[\hat{\Delta}_{\mathcal{R}}(B) > \hat{\Delta}_{\mathcal{R}}(B_\mu)] \leq \Pr[n \cdot \hat{\Delta}_{\mathcal{R}}(B) - \Delta_S(B) > \epsilon_2 \cdot OPT_\mu] \leq n^{-\ell'}/\binom{n}{k}$, where $\epsilon_2 = \epsilon - (1-1/e)\cdot\epsilon_1$. Here, the first inequality can be proved by Ineq. (6) and the definition of the *bad* set. The second inequality holds from the Chernoff bound and Ineq. (5). Because there are at most $\binom{n}{k}$ *bad* sets with k nodes, PRR-Boost returns a *bad* solution with probability at most $n^{-\ell'}$. Moreover, because $OPT_\mu \geq \mu(B^*)$, the probability that $\Delta_S(B_{sa}) < (1-1/e-\epsilon)\cdot\mu(B^*)$ is also upper bounded by $n^{-\ell'}$. The full proof can be found in the appendix. ■

Complexity. Lines 1-3 of PRR-Boost are essentially the IMM method with the goal of maximizing μ. Let EPT be the expected number of edges explored for generating a random PRR-graph, generating a random PRR-graph runs in $O(EPT)$ expected time. Denote the number of edges in a PRR-graph as the *size* of R, the expected size of a random PRR-graph is at most EPT. In the analysis, we also refer to the total size of PRR-graphs in $\mathcal{R}$ as the *size* of $\mathcal{R}$. By the analysis of the general IMM method, both the expected complexity of the sampling step in Line 2 and the size of $\mathcal{R}$ are $O(\frac{EPT}{OPT_\mu}\cdot(k+\ell)(n+m)\log n/\epsilon^2)$. The node selection step in Line 3 that maximizes μ corresponds to the greedy algorithm for maximum coverage, thus runs in time linear to the *size* of $\mathcal{R}$. Now, we analyze the node selection step in Line 4. After we insert a node into B, updating $\hat{\Delta}_{\mathcal{R}}(B\cup\{v\})$ for all $v \notin B\cup S$ takes time linear to the *size* of $\mathcal{R}$. Therefore, Line 4 runs in $O(\frac{EPT}{OPT_\mu}\cdot k(k+\ell)(n+m)\log n/\epsilon^2)$ expected time.

From Lemma 3-4, with a probability of at least $1-3n^{-\ell'} = 1-n^{-\ell}$, we have $\Delta_S(B_{\text{sa}}) \geq (1-1/e-\epsilon)\cdot\mu(B^*)$. Together with the above complexity analysis, we have the following theorem about PRR-Boost in Algorithm 2.

Theorem 2. *With a probability of at least $1-n^{-\ell}$, PRR-Boost returns a $(1-1/e-\epsilon)\cdot\frac{\mu(B^*)}{\Delta_S(B^*)}$-approximate solution. Moreover, it runs in $O(\frac{EPT}{OPT_\mu}\cdot k\cdot(k+\ell)(n+m)\log n/\epsilon^2)$ expected time.*

The approximation ratio given in Theorem 2 depends on the ratio of $\frac{\mu(B^*)}{\Delta_S(B^*)}$, which should be close to one if the lower bound function $\mu(B)$ is close to the actual boost of influence $\Delta_S(B)$, when $\Delta_S(B)$ is large. Section VI demonstrates that $\mu(B)$ is indeed close to $\Delta_S(B)$ in real datasets.

C. The PRR-Boost-LB Algorithm

PRR-Boost-LB is a simplification of PRR-Boost where we return the node set B_μ as the final solution. Recall that the estimation of μ only relies on the critical node set C_R of each boostable PRR-graph R. In the first phase of the PRR-graph generation, if we only need to obtain C_R, there is no need to explore incoming edges of a node v if $d_r[v] > 1$. Moreover, in the compression phase, we can obtain C_R right after computing $d_S[\cdot]$ and we can terminate the compression earlier. The sampling phase of PRR-Boost-LB usually runs faster than that of PRR-Boost, because we only need to generate C_R for each boostable PRR-graph R. In addition, the memory usage is significantly lower than that for PRR-Boost, because the averaged number of "critical nodes" in a random boostable PRR-graph is small in practice. In summary, compared with PRR-Boost, PRR-Boost-LB has the same approximation factor of $(1-1/e-\epsilon)\cdot\frac{\mu(B^*)}{\Delta_S(B^*)}$, but runs faster than PRR-Boost. We will compare PRR-Boost and PRR-Boost-LB by experiments in Section VI.

D. Discussion: The Budget Allocation Problem

A question one may raise is what is the best strategy if companies could freely decide how to allocation budget on both seeding and boosting. A heuristic method combing influence maximization algorithms and PRR-Boost is as follows. We could test different budget allocation strategy. For each allocation, we first identify seeds using any influence maximization algorithm, then we find boosted user by PRR-Boost. Finally, we could choose the budget allocation strategy leading to the largest boosted influence spread among all tested ones. In fact, the budget allocation problem could be much harder than the k-boosting problem itself, and its full treatment is beyond the scope of this study and is left as a future work.

VI. Experiments

We conduct extensive experiments using real social networks to test PRR-Boost and PRR-Boost-LB. Experimental results demonstrate their efficiency and effectiveness, and show their superiority over intuitive baselines. All experiments were conduct on a Linux machine with an Intel Xeon E5620@2.4GHz CPU and 30 GB memory. In PRR-Boost and PRR-Boost-LB, the generation of PRR-graphs and the estimation of objective functions are parallelized with OpenMP and executed using eight threads.

Table 1: Statistics of datasets and seeds (all directed)

Description	Digg	Flixster	Twitter	Flickr
number of nodes (n)	28 K	96 K	323 K	1.45 M
number of edges (m)	200 K	485 K	2.14 M	2.15 M
average influence probability	0.239	0.228	0.608	0.013
influence of 50 influential seeds	2.5 K	20.4 K	85.3 K	2.3 K
influence of 500 random seeds	1.8 K	12.5 K	61.8 K	0.8 K

Datasets. We use four real social networks: *Flixster* [28], *Digg* [29], *Twitter* [30], and *Flickr* [31]. All dataset have both directed social connections among its users, and actions of users with timestamps (e.g., rating movies, voting for stories, re-tweeting URLs, marking favorite photos). For all datasets, we learn the influence probabilities on edges using a widely accepted method proposed by Goyal et al. [32], remove edges with zero influence probability, and keep the largest weakly connected component. Table 1 summaries our datasets.

Boosted influence probabilities. To the best of our knowledge, no existing work quantitatively studies how influence among people changes respect to different kinds of "boosting strategies". Therefore, we assign the boosted influence probabilities as follows. For every edge e_{uv} with an influence probability of p_{uv}, let the boosted influence probability p'_{uv} be $1-(1-p_{uv})^\beta$ ($\beta > 1$). We refer to β as the *boosting parameter*. Due to the large number of combinations of parameters, we fix $\beta = 2$ unless otherwise specified. Intuitively, $\beta = 2$ indicates that every activated neighbor of a boosted node v has two independent chances to activate v. We also provide experiments showing the impacts of β.

Seed selection. We select seeds in two ways. (i) We use the IMM method [8] to select 50 influential nodes. In practice, the selected seeds typically correspond to highly influential customers selected with great care. Table 1 summaries the expected influence spread of selected seeds. (ii) We randomly select five sets of 500 seeds. The setting maps to the situation where some users become seeds spontaneously. Table 1 shows the average expected influence of five sets of selected seeds.

Baselines. As far as we know, no existing algorithm is applicable to the k-boosting problem. Thus, we compare our proposed algorithms with several heuristic baselines, as listed below.

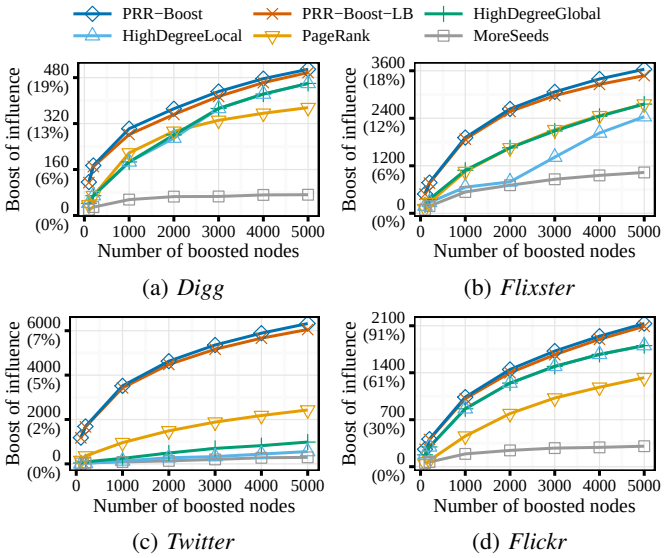
- *HighDegreeGlobal*: Starting from an empty set B, *HighDegreeGlobal* iteratively adds a node with the highest *weighted degree* to B, until k nodes are selected. We use four different definitions of the *weighted degree*. Suppose we have selected a set of nodes B, four definitions of the *weighted degree* of a node $u \notin (S \cup B)$ are as follows. (1) the sum of influence probabilities on outgoing edges (i.e., $\sum_{e_{uv}} p_{uv}$); (2) the "discounted" sum of influence probabilities on outgoing edges (i.e., $\sum_{e_{uv}, v \notin B} p_{uv}$); (3) the sum of the boost of influence probabilities on incoming edges (i.e., $\sum_{e_{vu}} [p'_{vu} - p_{vu}]$); (4) the "discounted" sum of the boost of influence probabilities on incoming edges (i.e., $\sum_{e_{vu}, v \notin B} [p'_{vu} - p_{vu}]$). Each definition outperforms the others in some experiments. We report the maximum boost of influence among four solutions as the result.
- *HighDegreeLocal*: The difference between *HighDegreeLocal* and *HighDegreeGlobal* is that, we first consider nodes close to seeds. We first try to select k nodes among neighbors of seeds. If there is not enough nodes to select, we continue to select among nodes that are two-hops away from seeds, and we repeat until k nodes are selected. We report the maximum boost of influence among four solutions selected using four definitions of the *weighted degree*.
- *PageRank*: We use the *PageRank* baseline for influence maximization problems [4]. When a node u has influence on v, it implies that node v "votes" for the rank of u. The transition probability on edge e_{uv} is $p_{vu}/\rho(u)$, where $\rho(u)$ is the summation of influence probabilities on all incoming edges of u. The restart probability is 0.15. We compute the *PageRank* iteratively until two consecutive iteration differ for at most 10^{-4} in L_1 norm.
- *MoreSeeds*: Given seed nodes and an integer k, we adapt the IMM framework to select k more seeds with the goal of maximizing the *increase* of the expected influence spread. We return the selected k seeds as the boosted nodes.

We do not compare our algorithms to the greedy algorithm with Monte-Carlo simulations. Because it has been shown to be extremely computationally expensive even for the classical influence maximization problem [1, 7].

Settings. For PRR-Boost and PRR-Boost-LB, we let $\epsilon = 0.5$ and $\ell = 1$ so that both algorithms return $(1-1/e-\epsilon)\cdot\frac{\mu(B^*)}{\Delta_S(B^*)}$-approximate solution with probability at least $1 - 1/n$. To enforce fair comparison, for all algorithms, we evaluate the boost of influence spread by $20,000$ Monte-Carlo simulations.

A. *Influential seeds*

In this part, we report results where the seeds are 50 influential nodes. The setting here maps to the real-world situation where the *initial adopters* are highly influential users selected with great care. We run each experiment five times and report the average results.



(a) *Digg* (b) *Flixster*

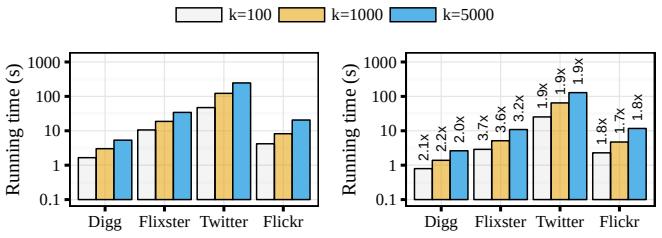
(c) *Twitter* (d) *Flickr*

Fig. 4: Boost of the influence versus k (influential seeds).

Quality of solution. Let B be the solution returned by an algorithm, the *quality of solution* or the *quality of boosted nodes* is the boost of the influence spread upon boosting B. Figure 4 compares the quality of solutions returned by different algorithms. PRR-Boost always return solutions with the highest quality, and PRR-Boost-LB returns solutions with slightly lower but comparable quality. Moreover, both PRR-Boost and PRR-Boost-LB outperform other baselines. In addition, we observe that *MoreSeeds* returns solutions with the lowest quality. This is because nodes selected by *MoreSeeds* are typically in the part of graph not covered by the existing seeds so that they could generate larger marginal influence. In contrast, boosting nodes should be selected close to existing seeds to make the boosting result more effective. Thus, our empirical result further demonstrates that k-boosting problem differs significantly from the influence maximization problem.

Running time. Figure 5 shows the running time of PRR-Boost and PRR-Boost-LB. For each dataset, the running time of both algorithm increases when k increases. This is mainly because the number of random PRR-graphs we have to generate increases when k increases. Figure 5 also shows that the running time is in general proportional to the number of nodes and edges for *Digg*, *Flixster* and *Twitter*, *but not for Flickr*. We observe that this is mainly because of the significantly smaller average influence probabilities on *Flickr* as shown in Table 1, and the accordingly significantly lower expected cost for generating a random PRR-graph (i.e., EPT) as we will show shortly in Table 3. In Figure 5, we also label the speedup of PRR-Boost-LB compared with PRR-Boost. PRR-Boost-LB is consistently faster than PRR-Boost in all our experiments. Together with Figure 4, we can see that PRR-Boost-LB is both efficient and effective. It returns solutions with quality comparable to PRR-Boost but runs faster. Because our approximation algorithms consistently outperform all heuristic methods with no performance guarantee in all tested cases,

we do not compare the running time of our algorithms with heuristic methods to avoid cluttering the results.



(a) PRR-Boost (b) PRR-Boost-LB

Fig. 5: Running time (influential seeds).

Effectiveness of the compression phase. Table 2 shows the "compression ratio" of PRR-graphs and memory usages of PRR-Boost and PRR-Boost-LB, demonstrating the importance of compressing PRR-graphs. The *compression ratio* is the ratio between the average number of uncompressed edges and average number of edges after compression in *boostable* PRR-graphs. Besides the total memory usage, we also show in parenthesis the memory usage for storing boostable PRR-graphs, which is measured as the additional memory usage starting from the generation of the first PRR-graph. For example, for the *Digg* dataset and $k = 100$, for boostable PRR-graphs, the average number of uncompressed edges is 1810.32, the average number of compressed edges is 2.41, and the compression ratio is 751.59. Moreover, the total memory usage of PRR-Boost is 0.07GB with around 0.01GB being used to storing "boostable" PRR-graphs. The compression ratio is high for two reasons. *First*, in practice, many nodes visited in the first phase cannot be reached by seed nodes. *Second*, among the remaining nodes, many of them can be merged into the super-seed node, and most non-super-seed nodes will be removed because they are not on any paths to the root node without going through the super-seed node. The high compression ratio and the memory used for storing compressed PRR-graphs show that the compression phase is an indispensable constituent of the generation of random PRR-graphs. For PRR-Boost-LB, the memory usage is *much lower* compared with PRR-Boost, because we only store "critical nodes" of boostable PRR-graphs. In our experiments with $\beta = 2$, on average, each boostable PRR-graph only has a few critical nodes, which explains the low memory usage of PRR-Boost-LB. If one is indifferent about the slightly difference between the quality of solutions returned by PRR-Boost-LB and PRR-Boost, we suggest to use PRR-Boost-LB because of its lower running time and lower memory usage.

Approximation factors in the *Sandwich Approximation*. Recall that the approximate ratio of PRR-Boost and PRR-Boost-LB depends on the ratio $\frac{\mu(B^*)}{\Delta_S(B^*)}$. The closer to one the ratio is, the better the approximation guarantee is. With B^* being unknown due to the NP-hardness of the problem, we show the ratio when the boost is relatively large. To obtain different boost sets with relatively large boosts, we generate 300 sets of k boosted nodes. The sets are constructed by replacing a random number of nodes in B_{sa} by other non-seed nodes, where B_{sa} is the solution returned by PRR-Boost. For a given B, we use PRR-graphs generated for finding B_{sa}

Table 2: Memory usage and compression ratio (influential seeds). Numbers in parentheses are additional memory usage for boostable PRR-graphs.

k	Dataset	PRR-Boost		PRR-Boost-LB
		Compression Ratio	Memory (GB)	Memory (GB)
100	*Digg*	1810.32 / 2.41 = 751.79	0.07 (0.01)	0.06 (0.00)
	Flixster	3254.91 / 3.67 = 886.90	0.23 (0.05)	0.19 (0.01)
	Twitter	14343.31 / 4.62 = 3104.61	0.74 (0.07)	0.69 (0.02)
	Flickr	189.61 / 6.86 = 27.66	0.54 (0.07)	0.48 (0.01)
5000	*Digg*	1821.21 / 2.41 = 755.06	0.09 (0.03)	0.07 (0.01)
	Flixster	3255.42 / 3.67 = 886.07	0.32 (0.14)	0.21 (0.03)
	Twitter	14420.47 / 4.61 = 3125.37	0.89 (0.22)	0.73 (0.06)
	Flickr	189.08 / 6.84 = 27.64	0.65 (0.18)	0.50 (0.03)

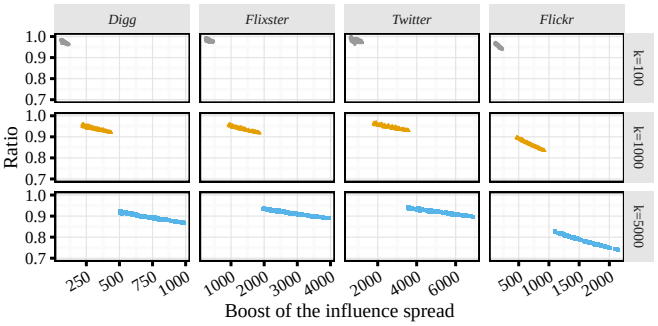
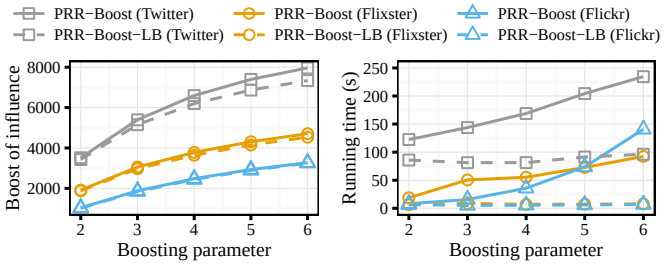


Fig. 6: Sandwich Approximation: $\frac{\mu(B)}{\Delta_S(B)}$ (influential seeds).

to estimate $\frac{\mu(B)}{\Delta_S(B)}$. Figure 6 shows the ratios for generated sets B as a function of $\Delta_S(B)$ for $k \in \{100, 1000, 5000\}$. Because we intend to show the ratio when the boost of influence is large, we do not show points corresponding to sets whose boost of influence is less than 50% of $\Delta_S(B_{sa})$. For all four datasets, the ratio is above 0.94, 0.83 and 0.74 for $k = 100, 1000, 5000$, respectively. For every dataset, we observe that the ratio is closer to one when k is smaller, and we now explain this. In practice, most boostable PRR-graphs have "critical nodes". When k is small, say 100, to utilize the *limited budget* k efficiently, PRR-Boost and PRR-Boost-LB tend to return node sets B so that every node in B is a critical node in many boostable PRR-graphs. For example, for *Twitter*, when $k = 100$, among PRR-graphs that have critical nodes and are activated upon boosting B_{sa}, above 98% of them have their critical nodes boosted (i.e., in B_{sa}). Meanwhile, many root node r of PRR-graphs without critical nodes may stay inactive. For a given PRR-graph R, if B contains critical nodes of R or if the root node of R stays inactive upon boosting B, $f_R^-(B)$ does not underestimate $f_R(B)$. Therefore, when k is smaller, the ratio of $\frac{\mu(B)}{\Delta_S(B)} = \frac{\mathbb{E}[f_R^-(B)]}{\mathbb{E}[f_R(B)]}$ tends to be closer to one. When k increases, we can boost more nodes, and root nodes of PRR-graphs without critical nodes may be activated, thus the approximation ratio tends to decrease. For example, for *Twitter*, when k increases from 100 to 5000, among PRR-graphs whose root nodes are activated upon boosting B_{sa}, the fraction of them having critical nodes decreases from around 98% to 88%. Accordingly, the ratio of $\mu(B_{sa})/\Delta_S(B_{sa})$ decreased by around 9% when k increases from 100 to 5000.

Effects of the boosted influence probabilities. In our experiments, we use the *boosting parameter* β to control the boosted influence probabilities on edges. The larger β is, the larger the



(a) *Boost of influence* (b) *Running time*

Fig. 7: Effects of the boosting parameter (influential seeds, $k = 1000$).

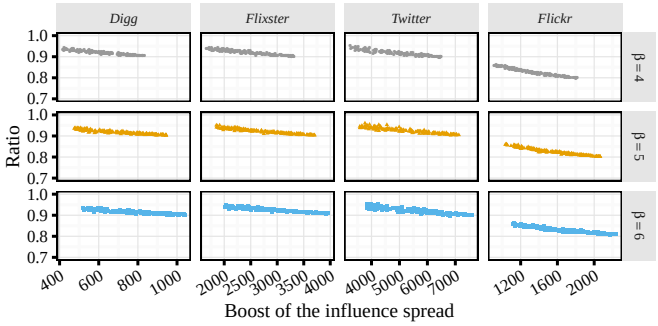
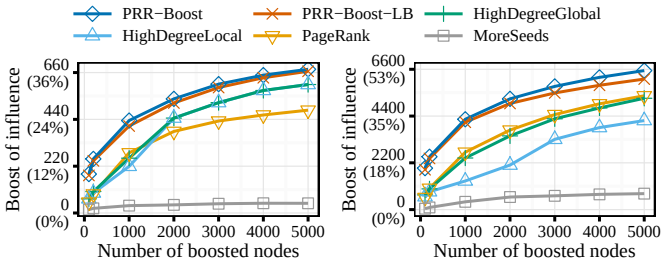


Fig. 8: Sandwich Approximation with varying boosting parameter: $\frac{\mu(B)}{\Delta_S(B)}$ (influential seeds, $k = 1000$).

boosted influence probabilities on edges are. Figure 7 shows the effects of β on the boost of influence and the running time, when $k = 1000$. For other values of k, the results are similar. In Figure 7a, the optimal boost increases when β increases. When β increases, for *Flixster* and *Flickr*, PRR-Boost-LB returns solution with quality comparable to those returned by PRR-Boost. For *Twitter*, we consider the slightly degenerated performance of PRR-Boost-LB acceptable because PRR-Boost-LB runs significantly faster. Figure 7b shows the running time for PRR-Boost and PRR-Boost-LB. When β increases, the running time of PRR-Boost increases accordingly, but the running time of PRR-Boost-LB remains almost unchanged. Therefore, compared with PRR-Boost, PRR-Boost-LB is more scalable to *larger boosted influence probabilities on edges*. In fact, when β increases, a random PRR-graph tends to include more nodes and edges. The running time of PRR-Boost increases mainly because the cost for PRR-graph generation increases. However, when β increases, we observe that the cost for obtaining "critical nodes" for a random PRR-graph does not change much, thus the running time of PRR-Boost-LB remains almost unchanged. Figure 8 shows the approximation ratio of the sandwich approximation strategy with varying boosting parameters. We observe that, for every dataset, when we increase the boosting parameter, the ratio of $\frac{\mu(B)}{\Delta_S(B)}$ for large $\Delta_S(B)$ remains almost the same. This suggests that both our proposed algorithms remain effective when we increase the boosted influence probabilities on edges.

B. Random seeds

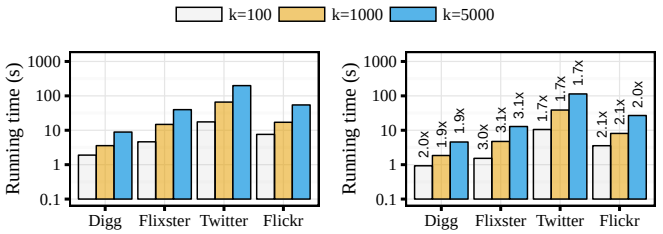
In this part, we select five sets of 500 random nodes as seeds for each dataset. The setting here maps to the real



(a) *Digg* (b) *Flixster*

(c) *Twitter* (d) *Flickr*

Fig. 9: Boost of the influence versus k (random seeds).



(a) PRR-Boost (b) PRR-Boost-LB

Fig. 10: Running time (random seeds).

situation where some users become *seeds* spontaneously. All experiments are conducted on five sets of random seeds, and we report the average results.

Quality of solution. We select up to 5000 nodes and compare our algorithms with baselines. From Figure 9, we can draw conclusions similar to those drawn from Figure 4 where the seeds are highly influential users. Both PRR-Boost and PRR-Boost-LB outperform all baselines.

Running time. Figure 10 shows the running time of PRR-Boost and PRR-Boost-LB, and the speedup of PRR-Boost-LB compared with PRR-Boost. Figure 10b shows that PRR-Boost-LB runs up to three times faster than PRR-Boost. Together with Figure 9, PRR-Boost-LB is in fact both efficient and effective given randomly selected seeds.

Effectiveness of the compression phase. Table 3 shows the compression ratio of PRR-Boost, and the memory usage of both proposed algorithms. Given randomly selected seed nodes, the compression step of PRR-graphs is also very effective. Together with Table 2, we can conclude that the compression phase is an indispensable step for both cases where the seeds are highly influence users or random users.

Approximation factors in the *Sandwich Approximation*. The approximate ratio of PRR-Boost and PRR-Boost-LB depends on the ratio $\frac{\mu(B^*)}{\Delta_S(B^*)}$. We use the same method to generate

Table 3: Memory usage and compression ratio (random seeds).

k	Dataset	PRR-Boost		PRR-Boost-LB
		Compression Ratio	Memory (GB)	Memory (GB)
100	*Digg*	3069.15 / 5.61 = 547.28	0.07 (0.01)	0.06 (0.00)
	Flixster	3754.43 / 25.83 = 145.37	0.24 (0.06)	0.19 (0.01)
	Twitter	16960.51 / 56.35 = 300.96	0.78 (0.11)	0.68 (0.01)
	Flickr	701.84 / 18.12 = 38.73	0.56 (0.09)	0.48 (0.01)
5000	*Digg*	3040.94 / 5.59 = 544.19	0.12 (0.06)	0.07 (0.01)
	Flixster	3748.74 / 25.86 = 144.94	0.71 (0.53)	0.21 (0.03)
	Twitter	16884.86 / 57.29 = 294.72	1.51 (0.84)	0.72 (0.05)
	Flickr	701.37 / 18.10 = 38.75	1.00 (0.53)	0.50 (0.03)

different sets of boosted nodes B as in the previous sets of experiments. Figure 11 shows the ratios for generated sets B as a function of $\Delta_S(B)$ for $k \in \{100, 1000, 5000\}$. For all four datasets, the ratio is above 0.76, 0.62 and 0.47 for $k = 100, 1000, 5000$, respectively. As from Figure 6, the ratio is closer to one when k is smaller. Compared with Figure 6, we observe that the ratios in Figure 11 are lower. The main reason is that, along with many PRR-graphs with critical nodes, many PRR-graphs without critical nodes are also boosted. For example, for *Twitter*, when $k = 5000$, among PRR-graphs whose root nodes are activated upon boosting B_{sa}, around 25% of them do not have critical nodes, and around 3% of them have critical nodes but their critical nodes are not in B_{sa}. Note that, although the approximation guarantee of our proposed algorithms decreases as k increases, Figure 9 shows that our proposed algorithms still outperform all other baselines.

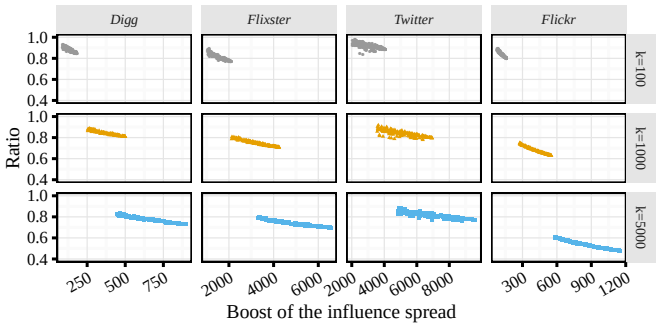
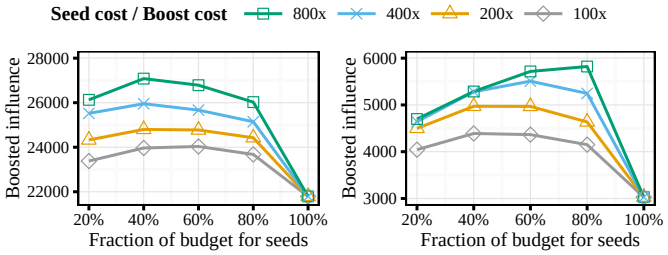


Fig. 11: Sandwich Approximation: $\frac{\mu(B)}{\Delta_S(B)}$ (random seeds).

C. Budget allocation between seeding and boosting

In this part, we vary both the number of seeders and the number of boosted nodes. Under the context of viral marketing, this corresponds to the situation where a company can decide both the number of free samples and the number of coupons they offer. Intuitively, targeting a user as a seeder (e.g., offering a free product and rewarding for writing positive opinions) must cost more than boosting a user (e.g., offering a discount or displaying ads). In the experiments, we assume that we can target 100 users as seed nodes with all the budget. Moreover, we assume that targeting a seeder costs 100 to 800 times as much as boosting a user. For example, suppose targeting a seeder costs 100 times as much as boosting a user: we can choose to spend 20% of our budget on targeting initial adopters (i.e., finding 20 seed users and boosting 8000 users); or, we can spend 80% of the budget on targeting initial adopters (i.e, finding 80 seeds and boosting 2000 users). We explore how the expected influence spread changes, when we decrease the number of seed users and increase the number of boosted users. Given the budget allocation (i.e., the number of seeds and the number boosted users), we first identify a set of influential seeds using the `IMM` method, then we use `PRR-Boost` to select the set of nodes we boost. Finally, we use $20,000$ Monte-Carlo simulations to estimate the expected boosted influence spread.



(a) *Flixster* (b) *Flickr*

Fig. 12: Budget allocation between seeding and boosting.

Figure 12 shows the results for *Flixster* and *Flickr*. We observe that the boosted influence spread with a mixed budget among initial adopters and boosting users achieves higher final influence spread than spending all budget on initial adopters. For example, for cost ratio of 800 between seeding and boosting, if we choose 80% budget for seeding and 20% for boosting, we would achieve around 20% and 92% higher influence spread than pure seeding, for *Flixster* and *Flickr* respectively. Moreover, the best budget mix is different for different networks and different cost ratio, suggesting the need for specific tuning and analysis for each case.

VII. Conclusion

In this work, we address a novel k-boosting problem that asks how to *boost* the influence spread by offering k users incentives so that they are more likely to be influenced by friends. We develop efficient approximation algorithms, `PRR-Boost` and `PRR-Boost-LB`, that have *data-dependent* approximation factors. Both `PRR-Boost` and `PRR-Boost-LB` are delicate integration of *Potentially Reverse Reachable Graphs* and the state-of-the-art techniques for influence maximization problems. We conduct extensive experiments on real datasets using our proposed algorithms. In our experiments, we consider both the case where the seeds are highly influential users, and the case where the seeds are randomly selected users. Results demonstrate the superiority of our proposed algorithms over intuitive baselines. Compared with `PRR-Boost`, experimental results show that `PRR-Boost-LB` returns solution with comparable quality but has significantly lower computational costs. In our experiments, we also explore the scenario where we are allowed to determine how to spend the limited budget on both targeting initial adopters and boosting users. Experimental results demonstrate the importance of studying the problem of targeting initial adopters and boosting users with a mixed strategy. We leave the full exploration of this topic as our future work.

REFERENCES

[1] D. Kempe, J. Kleinberg, and É. Tardos, "Maximizing the spread of influence through a social network," in *KDD*, 2003.

[2] T. Carnes, C. Nagarajan, S. M. Wild, and A. Van Zuylen, "Maximizing influence in a competitive social network: a follower's perspective," in *ICEC*, 2007.

[3] W. Chen, Y. Wang, and S. Yang, "Efficient influence maximization in social networks," in *KDD*, 2009.

[4] W. Chen, C. Wang, and Y. Wang, "Scalable influence maximization for prevalent viral marketing in large-scale social networks," in *KDD*, 2010.

[5] W. Chen, Y. Yuan, and L. Zhang, "Scalable influence maximization in social networks under the linear threshold model," in *ICDM*, 2010.

[6] C. Borgs, M. Brautbar, J. Chayes, and B. Lucier, "Maximizing social influence in nearly optimal time," in *SODA*, 2014.

[7] Y. Tang, X. Xiao, and Y. Shi, "Influence maximization: Near-optimal time complexity meets practical efficiency," in *SIGMOD*, 2014.

[8] ——, "Influence maximization in near-linear time: A martingale approach," in *SIGMOD*, 2015.

[9] C. Budak, D. Agrawal, and A. El Abbadi, "Limiting the spread of misinformation in social networks," in *WWW*, 2011.

[10] X. He, G. Song, W. Chen, and Q. Jiang, "Influence blocking maximization in social networks under the competitive linear threshold model." in *SDM*, 2012.

[11] "Global trust in advertising," http://www.nielsen.com/us/en/insights/reports/2015/global-trust-in-advertising-2015.html, accessed: 2016-09-18.

[12] J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. VanBriesen, and N. Glance, "Cost-effective outbreak detection in networks," in *ICDM*, 2007.

[13] A. Goyal, W. Lu, and L. V. S. Lakshmanan, "Celf++: optimizing the greedy algorithm for influence maximization in social networks," in *WWW*, 2011.

[14] W. Chen, L. V. S. Lakshmanan, and C. Castillo, "Information and influence propagation in social networks," *Synthesis Lectures on Data Management*, 2013.

[15] H. T. Nguyen, T. N. Dinh, and M. T. Thai, "Stop-and-stare: Optimal sampling algorithms for viral marketing in billion-scale networks," in *SIGMOD*, 2016.

[16] V. Chaoji, S. Ranu, R. Rastogi, and R. Bhatt, "Recommendations to boost content spread in social networks," *IW3C2*, 2012.

[17] D.-N. Yang, H.-J. Hung, W.-C. Lee, and W. Chen, "Maximizing acceptance probability for active friending in online social networks," in *KDD*, 2013.

[18] S. Antaris, D. Rafailidis, and A. Nanopoulos, "Link injection for boosting information spread in social networks," *SNAM*, vol. 4, no. 1, 2014.

[19] D. Rafailidis, A. Nanopoulos, and E. Constantinou, "With a little help from new friends: Boosting information cascades in social networks based on link injection," *Journal of Systems and Software*, vol. 98, 2014.

[20] D. Rafailidis and A. Nanopoulos, "Crossing the boundaries of communities via limited link injection for information diffusion in social networks," in *WWW Companion*, 2015.

[21] Y. Yang, X. Mao, J. Pei, and X. He, "Continuous influence maximization: What discounts should we offer to social network users?" in *SIGMOD*, 2016.

[22] W. Lu, W. Chen, and L. V. S. Lakshmanan, "From competition to complementarity: Comparative influence diffusion and maximization," *VLDB Endow.*, vol. 9, no. 2, 2015.

[23] W. Chen, F. Li, T. Lin, and A. Rubinstein, "Combining traditional marketing and viral marketing with amphibious influence maximization," in *EC*, 2015.

[24] R. M. Karp, *Reducibility among combinatorial problems*. Springer, 1972.

[25] L. G. Valiant, "The complexity of enumeration and reliability problems," *SIAM Journal on Computing*, vol. 8, no. 3, 1979.

[26] Y. Lin, W. Chen, and J. C. S. Lui, "Boosting information spread: An algorithmic approach," *arXiv:1602.03111 [cs.SI]*, 2016.

[27] Y. Lin and J. C. S. Lui, "Analyzing competitive influence maximization problems with partial information: An approximation algorithmic framework," *Performance Evaluation*, vol. 91, 2015.

[28] M. Jamali and M. Ester, "A matrix factorization technique with trust propagation for recommendation in social networks," in *RecSys*, 2010.

[29] T. Hogg and K. Lerman, "Social dynamics of digg," *EPJ Data Science*, vol. 1, no. 1, 2012.

[30] N. O. Hodas and K. Lerman, "The simple rules of social contagion," *Scientific reports*, vol. 4, 2014.

[31] M. Cha, A. Mislove, and K. P. Gummadi, "A measurement-driven analysis of information propagation in the flickr social network," in *Proceedings of the 18th international conference on World wide web*. ACM, 2009.

[32] A. Goyal, F. Bonchi, and L. V. S. Lakshmanan, "Learning influence probabilities in social networks," in *WSDM*, 2010.

APPENDIX

Lemma 4. *Given that Ineq. (5)-(6) hold, with a probability of at least* $1-n^{-\ell'}$*, we have*

$$\Delta_S(B_{sa}) \geq (1-1/e-\epsilon)\cdot OPT_\mu \geq (1-1/e-\epsilon)\cdot \mu(B^*).$$

Proof: Let B be a boost set with k nodes, we say that B is a *bad* set if $\Delta_S(B) < (1-1/e-\epsilon)\cdot OPT_\mu$. To prove Lemma 4, we first show that each *bad* boost set with k nodes is returned by Algorithm 2 with a probability of at most $(1-n^{-\ell'})/\binom{n}{k}$. Let B be an arbitrary *bad* set with k nodes, we have

$$\Delta_S(B) < (1-1/e-\epsilon)\cdot OPT_\mu. \tag{7}$$

If we return B as the B_{sa}, we must have $\hat{\Delta}_{\mathcal{R}}(B) > \hat{\Delta}_{\mathcal{R}}(B_\mu)$. Therefore, the probability that B is returned is upper bounded by $\Pr[\hat{\Delta}_{\mathcal{R}}(B) > \hat{\Delta}_{\mathcal{R}}(B_\mu)]$. From Ineq. (6)-(7), we have

$$\begin{aligned}
&n\cdot\hat{\mu}_{\mathcal{R}}(B_\mu) - \Delta_S(B)\\
\geq&(1-1/e)\cdot(1-\epsilon_1)\cdot OPT_\mu - (1-1/e-\epsilon)\cdot OPT_\mu\\
=&(\epsilon-(1-1/e)\cdot\epsilon_1)\cdot OPT_\mu.
\end{aligned}$$

Let $\epsilon_2 = \epsilon-(1-1/e)\cdot\epsilon_1$, we have

$$\begin{aligned}
&\Pr[\hat{\Delta}_{\mathcal{R}}(B) > \hat{\Delta}_{\mathcal{R}}(B_\mu)] \leq \Pr[\hat{\Delta}_{\mathcal{R}}(B) > \hat{\mu}_{\mathcal{R}}(B_\mu)]\\
\leq&\Pr[n\cdot\hat{\Delta}_{\mathcal{R}}(B) - \Delta_S(B) > n\cdot\hat{\mu}_{\mathcal{R}}(B_\mu) - \Delta_S(B)]\\
\leq&\Pr[n\cdot\hat{\Delta}_{\mathcal{R}}(B) - \Delta_S(B) > \epsilon_2\cdot OPT_\mu].
\end{aligned}$$

Let $p = \Delta_S(B)/n$, we know $p = \mathbb{E}[f_R(B)]$ from Lemma 1. Recall that $\theta = |\mathcal{R}|$ and $\hat{\Delta}_{\mathcal{R}}(B) = \big(\sum_{R\in\mathcal{R}} f_R(B)\big)/\theta$, we have

$$\begin{aligned}
&\Pr\left[n\cdot\hat{\Delta}_{\mathcal{R}}(B) - \Delta_S(B) > \epsilon_2\cdot OPT_\mu\right]\\
=&\Pr\left[\sum_{R\in\mathcal{R}} f_R(B) - \theta p > \frac{\epsilon_2\cdot OPT_\mu}{np}\cdot\theta p\right].
\end{aligned}$$

Let $\delta = \frac{\epsilon_2\cdot OPT_\mu}{np}$, by Chernoff bound, we have

$$\begin{aligned}
&\Pr\Big[\sum_{R\in\mathcal{R}} f_R(B) - \theta p > \frac{\epsilon_2\cdot OPT_\mu}{np}\cdot\theta p\Big]\\
\leq&\exp\Big(-\frac{\delta^2}{2+\delta}\cdot\theta p\Big) = \exp\Big(-\frac{\epsilon_2^2\cdot OPT_\mu^2}{2n^2p+\epsilon_2\cdot OPT_\mu\cdot n}\cdot\theta\Big)\\
\leq&\exp\Big(-\frac{\epsilon_2^2\cdot OPT_\mu^2}{2n(1-1/e-\epsilon)\cdot OPT_\mu+\epsilon_2\cdot OPT_\mu\cdot n}\cdot\theta\Big) \quad \text{(Ineq. (7))}\\
\leq&\exp\Big(-\frac{(\epsilon-(1-1/e)\cdot\epsilon_1)^2\cdot OPT_\mu}{n\cdot(2-2/e)}\cdot\theta\Big)\\
\leq&\exp\Big(-\log\big(\tbinom{n}{k}\cdot(2n^{\ell'})\big)\Big) \quad \text{(Ineq. (5))}\\
\leq&n^{-\ell'}/\tbinom{n}{k}.
\end{aligned}$$

Because there are at most $\binom{n}{k}$ *bad* sets B with k nodes, by union bound, the probability that Algorithm 2 returns a *bad* solution is at most $n^{-\ell'}$. Because $OPT_\mu \geq \mu(B^*)$, the probability that Algorithm 2 returns a solution so that $\Delta_S(B_{sa}) < (1-1/e-\epsilon)\cdot\mu(B^*)$ is also at most $n^{-\ell'}$. ■