Where & Who Should You Advertise? Influence Maximization for Two-Layer Networks

Yishi Lin
The Chinese University of Hong Kong
yishilin14@gmail.com

John C.S. Lui
The Chinese University of Hong Kong
cslui@cse.cuhk.edu.hk

ABSTRACT

"*Where to advertise*" asks how to select content providers for displaying advertisements so to reach large audiences. "*Who to advertise*", on the other hand, asks how to identify influential users so to spread marketing messages far and wide in a social network. The *Amphibious Influence Maximization* (AIM) problem asks a combination of theses two questions. To be specific, it asks how to select b_c seed content providers and b_v seed users so that the influence could spread widely from seed content providers through seed users to the entire social network. Approximating the AIM problem to within any constant factor is NP-hard, and no existing method scales to large social networks. To address the scalability issue, we present two algorithms AIM-α and AIM-$\emptyset$. For $\epsilon > 0$, AIM-α returns seeds and an approximation ratio $(1 - 1/e - \epsilon) \cdot \rho$ where $\rho \geq \alpha/b_c$. The returned solution is a $(1 - 1/e - \epsilon) \cdot \rho$-approximate solution with high probability. In AIM-α, the parameter α $(1 \leq \alpha \leq b_c)$ controls the trade off between the approximation ratio and the efficiency. The other algorithm AIM-$\emptyset$ is an efficient heuristic algorithm. It is a condensed version of AIM-α. We conduct extensive experiments to evaluate and compare AIM-α and AIM-$\emptyset$. In a dataset with 1.7 million users and 104 thousands content providers, when $\epsilon = 0.5$, AIM-α where $\alpha = 1$ finds 10 seed content providers and 1000 seed users within 72 seconds, and AIM-$\emptyset$ finishes within 16 seconds. Moreover, our experiments show that AIM-α and AIM-$\emptyset$ always return solution with similar influence spread.

1 INTRODUCTION

With the popularity of online social networks, *viral marketing* has become an important channel for advertisers to achieve marketing objectives. In real marketing campaigns, advertisers have to consider not just "*who to advertise to*" but also "*where to advertise*". For example, to promote a local business such as a coffee shop, shop owners must decide which social media or blogging sites they should reach out to (i.e., where to advertise). Meanwhile, to make sure their marketing messages spread far and wide among consumers, they also have to target influential coffee lovers and display their advertorials to them (i.e., who to advertise to). Another example would be how to open pop-up stores so to increase brand awareness and boost sales. In this case, retailers have to simultaneously decide where to open their stores and who are influential consumers they should attract.

The "*who to advertise to*" part is in fact the classical *Influence Maximization* (IM) problem [14] that has been intensively studied. Adding the question of "*where to advertise*" makes the classical problem more practical. Recently, Chen et al. [6] formulated a combination of these two questions as the *Amphibious Influence Maximization* (AIM) problem. In AIM, we are given a two-layer network. The first layer contains a set C of "content providers" (e.g., blogs, potential locations for stores). The second layer is a social network modeled by a directed graph $G = (V, E)$, where V represents users and E represents edges. A bipartite graph $B = (C, V, M)$ models the inter-layer links. Here, M is a $|C| \times |V|$ matrix where M_{cv} represents the "influence probability" from content provider c to user v. Influence propagates first from "seed content providers" to "seed users", then to other users in the social network following a predetermined influence propagation model. The AIM problem asks to select b_c seed content providers and b_v seed users so that the expected influence spread upon seeding them is maximized.

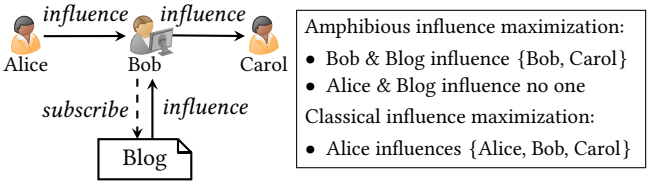


Figure 1: Amphibious influence maximization versus the classical influence maximization. The social network contains three users. The "Blog" is a content provider.

Figure 1 provides an example of AIM. In the social network, Alice influences Bob, and Bob influences Carol. There is one "content provider", a blog. Bob subscribes to this blog, and is influenced by its articles. In IM, if Alice is a seed user, all users are influenced. However, in AIM, if we want to influence any user, we should simultaneously seed the blog and Bob. The blog and Bob together influence two users (i.e., Bob and Carol), thus have an influence spread of two. This example shows the difference between IM and AIM. It is important to note that influential users in the IM problem may not be good seed users in the AIM problem.

The AIM problem is important yet challenging. Chen et el. [6] proved that *it is NP-hard to even approximate AIM to within any constant factor*. They proposed the *Sampled Double Greedy* (SDG) algorithm that returns a $(1 - 1/e - \epsilon)^3$ solution with a probability of at least $1 - \delta$, for any $0 < \epsilon, \delta < 1$. The time complexity is $O(|V|^r \log |C|^r (b_c + b_v)(|C| + |V| + |M| + |E|)\epsilon^{-(r+2)} \log(1/\delta))$, where r is the rank of M and $|M|$ is the number of non-zero entries in M. Unfortunately, the running time of SDG grows exponentially with the rank r of the matrix M, thus it cannot scale to large networks with arbitrary form of M.

Contributions. To conduct efficient amphibious influence maximization in real marketing campaigns, it is essential to have practical algorithms that work for the AIM problem in general (i.e., has no restriction on the rank of M). Motivated by this, we present two new algorithms for the AIM problem in this paper.

The first algorithm AIM-α returns a $(1-1/e-\epsilon)\cdot\alpha/b_c$-approximate solution with high probability. The integer parameter α controls the trade off between the approximate ratio and the efficiency: AIM-1 is the most efficient one, AIM-b_c is the most effective one. Together with seed content providers and seed users, AIM-α also returns an approximation ratio $(1-1/e-\epsilon)\cdot\rho$ in each execution, where ρ is always at least α/b_c. The obtained set of seeds is a $(1-1/e-\epsilon)\cdot\rho$-approximate solution with high probability. AIM-α provides approximation guarantee better than SDG if $\alpha/b_c \geq (1-1/e-\epsilon)^2$ (e.g., $\alpha \geq b_c/58$ when $\epsilon = 0.5$). We believe that the budget b_c in a real marketing campaign is usually limited, say only a few. For example, retailers usually open pop-up stores in a few highly selective locations because opening each store incurs high costs. Under the assumption of small b_c, AIM-α with small α would provide performance guarantee better than the unscalable SDG.

The second algorithm AIM-$\emptyset$ is an efficient heuristic algorithm. It returns seeds with similar influence spread as compared to AIM-α in our experiments. It is particularly useful when we have to maintain seeds in a dynamic two-layer network. In this case, we can first compare solutions returned by AIM-α and AIM-$\emptyset$ in the initial network. Suppose they return solution with similar influence spread, when there are minor changes to the network, it is very likely that they still return solution with similar influence spread. Thus, we could update seeds via the *much more efficient* AIM-$\emptyset$ algorithm.

To evaluate and compare the performance of proposed algorithms, we conduct extensive experiments using real social networks and bipartite graphs constructed in various ways. Our experiments show that both of our algorithms are efficient and effective. For example, let $\epsilon = 0.5$ and $\delta = 0.001$. In a dataset with 1.7 million users, 22.6 million edges among users, 104 thousands content providers and 8.5 million edges between users and providers, when $\epsilon = 0.5$, AIM-α where $\alpha = 1$ finds 10 seed content providers and 1000 seed users within 72 seconds, and AIM-$\emptyset$ finishes within 16 seconds. Our experiments show that AIM-1 is both efficient and effective. Moreover, AIM-$\emptyset$ is more efficient than AIM-1, and it returns solution with comparable influence spread as compared to AIM-1.

Paper organization. Section 2 gives background and preliminaries. Section 3 introduces the *Amphibious Influence Maximization* (AIM) problem. We present our basic algorithm in Section 4, and AIM-α built upon it in Section 5. We present a heuristic method AIM-$\emptyset$ in Section 6. We show experimental results in Section 7. Section 8 provides related works. Section 9 concludes the paper.

2 BACKGROUND AND RELATED WORK

We first introduce the classical influence maximization (IM) problem. Then, we review state-of-the-art IM techniques, which are preliminaries of this paper.

2.1 Influence maximization

Kempe et al. [14] first formulated the *Influence Maximization* problem as a discrete optimization problem. They also proposed the *Independent Cascade* (IC) model and the *Linear Threshold* (LT) model to describe how influence propagates in a social network. In these models, social networks are modeled by a directed graph $G = (V, E)$ with n nodes and m edges. Every edge $e_{uv} \in E$ is associated with an influence probability p_{uv}. Let $S \subseteq V$ be a set of seed users. Initially, nodes in S are influenced. Under the IC model, when a node u gets influenced, it tries to influence each of its uninfluenced neighbor v and succeeds with probability p_{uv}. Under the LT model, each node v randomly determines a threshold $\theta_v \sim U[0,1]$. Then, an uninfluenced node v gets influenced once the total influence probabilities of his influenced neighbors exceeds θ_v (i.e., $\sum_{u:\text{influenced neighbor of } v} p_{uv} \geq \theta_v$). The propagation process under either model continues until no new nodes gets influenced. Given a seed set $S \subseteq V$, the expected influence spread $\sigma(S)$ is the expected number of nodes influenced at the end of the propagation.

Under a predetermined propagation model, the *Influence Maximization* (IM) problem asks how to select a set $S \subseteq V$ of k seed nodes so that the expected influence spread is maximized. Formally, we want to find

$$S = \underset{S\subseteq V, |S|=k}{\arg\max}\ \sigma(S). \tag{1}$$

Under both the IC and LT model, the IM problem is NP-hard [14], and computing $\sigma(\cdot)$ is #P-hard [7, 8]. Due to the NP-hardness of the IM problem, there are many studies focusing on finding efficient approximate algorithms [4, 10, 11, 13, 17, 21, 26, 27, 29].

2.2 State-of-the-art IM techniques

We now introduce state-of-the-art influence maximization algorithms. They are based on the *Reverse Reachable Sets* (RR-sets) [4].

RR-sets. Suppose influence propagates in social networks under a predetermined model. 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 S influences r in a random diffusion process. Note that the definition of RR-set here is in fact the definition of *Reverse Influence Set* (RI-set) [27], a generalized definition of the original RR-set. A random RR-set is generated according to a random "root" node r. By definition, we have $\sigma(S) = n\cdot\mathbb{E}_R[\mathbb{I}(R\cap S\neq\emptyset)]$ for any $S \subseteq V$, where $\mathbb{I}(\cdot)$ is the indicator function. By Chernoff bound, if we have a sufficiently large collection $\mathcal{R}$ of random RR-sets, the value of $n/|\mathcal{R}|\cdot\sum_{R\in\mathcal{R}}\mathbb{I}(R\cap S\neq\emptyset)$ would be a good estimation of $\sigma(S)$. The detailed generation of RR-sets is beyond the scope of this paper. We refer interested reader to [4] and [26] for more details.

IM algorithms. Prior art includes IMM (Influence Maximization via Martingale) [27], SSA (Stop-and-Stare Algorithm) and its dynamic version D-SSA [21]. In these methods, estimating the influence spread is done by sampling a sufficiently large number of random RR-sets. Based on the estimated influence spread, a greedy *seed selection* subroutine selects and returns k seed nodes as the solution. We summarize the main results of these methods as follows.

Theorem 2.1 (IMM, SSA, and D-SSA). *Suppose we are given a social network $G = (V, E)$, an integer k, and parameters $0 < \epsilon, \delta < 1$, and we know how to generate random RR-sets. Let the optimal seed set be S_k^*. Denote the influence spread estimated by sampled RR-sets by $\hat{\sigma}(\cdot)$. With probability at least $1-\delta$, the influence maximization algorithm returns a solution $\hat{S}_k$ satisfying $(1+\epsilon_a)\cdot\sigma(\hat{S}_k) > \hat{\sigma}(\hat{S}_k)$ and $\hat{\sigma}(S_k^*) \geq (1-\epsilon_b)\cdot\sigma(S_k^*)$, where $(1-1/e)(\epsilon_a+\epsilon_b)/(1+\epsilon_a) \leq \epsilon$. Conditioned on the above two inequalities , we have $\sigma(\hat{S}_k) \geq (1-1/e-\epsilon)\cdot\sigma(S_k^*)$.*

Table 1: Frequently used notations.

Notation	Description
G=(V, E)	a social network with users V and edges E
B=(C, V, M)	a bipartite graph between C and V (M is a $\|C\| \times \|V\|$ probability matrix)
n, n_c	# of users, # of content providers
b_c, b_v	the budget for seeding content providers / users
X, Y	seed content providers / users ($X \subseteq C, Y \subseteq V$)
$\sigma(X, Y)$	the expected influence spread of (X, Y)
OPT	the optimal influence spread
R	a random RR-set ($R \subseteq V$)
R_X	a random RR$_X$-set ($X \subseteq C, R_X \subseteq V$)
R_Y	a random RR$_Y$-set ($Y \subseteq V, R_Y \subseteq C$)

Thus, these methods return a $(1 - 1/e - \epsilon)$-approximate solution with probability at least $1 - \delta$.

Parameters ϵ_a and ϵ_b in Theorem 2.1 are determined by the specific IM algorithm. In IMM, they are functions of n, k, ϵ and δ. In SSA, the authors provided several suggested settings for different sizes of the input network. D-SSA determines values dynamically at runtime. The IMM method runs in $O((k + \log(1/\delta))(n + m) \log n/\epsilon^2)$ expected time. In practice, SSA and D-SSA are usually more efficient than IMM especially when the number of seeds k is large.

3 PROBLEM DEFINITION

In this section, we present the formal definition of the *Amphibious Influence Maximization* problem and its inapproximability result [6]. Then, we review the existing solution and discuss its limitation. Table 1 summarizes frequently used notations.

Amphibious influence propagation model. Influence propagates in a two-layer network. The first layer contains a set C of content providers. Let $n_c = |C|$ be the number of content providers. The second layer is a directed social network $G = (V, E)$. Let $n = |V|$ be the number of users. A bipartite graph $B = (C, V, M)$ models the inter-layer links. Here, M is a $n_c \times n$ matrix where M_{cv} is the probability that the content provider c influences a user v. We let $M_{cv} = 0$ if c and v are not connected. Let $X \subseteq C$ be a set of seed content providers and let $Y \subseteq V$ be a set of seed users, influence propagates as follows. First, every seed content provider $c \in X$ tries to influence every seed user $v \in Y$ and succeeds with probability M_{cv}. It is easy to see that a seed user $v \in Y$ gets influenced in this step with probability $1 - \prod_{c \in X}(1 - M_{cv})$. Let $Y' \subseteq Y$ be the set of influenced seed users. In the second step, nodes in Y' act as seed users as in the classical influence propagation model (e.g., IC or LT) and spread their influence. We use $\sigma(X, Y)$ to denote the expected influence spread of (X, Y). By definition of the propagation model, we have $\sigma(X, Y) = \sum_{Y' \subseteq Y} \Pr[X \text{ influences } Y'] \cdot \sigma(Y')$, where $\sigma(Y')$ is the influence spread of $Y' \subseteq Y$ in the social network as in the classical influence propagation model. Given X (resp. Y), $\sigma(X, Y)$ is a monotone and submodular function of Y (resp. X) [6].

In this paper, when the context is clear, we use *seeds* or *solution* to refer to a set of seeds containing *both* seed content providers and seed users. Moreover, we assume that influence propagates among users under a predetermined model with known methods to generate RR-sets (e.g., IC or LT).

Amphibious Influence Maximization (AIM) problem. The AIM problem is formally defined as follows [6].

Definition 3.1 (Amphibious Influence Maximization). Given a bipartite graph $B = (C, V, M)$, a social network $G = (V, E)$, and budgets b_c and b_v, find a subset $X \subseteq C$ of size b_c and a subset $Y \subseteq V$ of size b_v such that the influence spread $\sigma(X, Y)$ is maximized. That is, finding X and Y such that

$$(X, Y) = \arg\max_{X \subseteq C, |X| = b_c, Y \subseteq V, |Y| = b_v} \sigma(X, Y). \quad (2)$$

Inapproximability result. Seed content providers spread influence to non-seed users *through* seed users. The hardness of AIM lies in simultaneously identifying a good combination of these two types of seeds. In fact, Chen et al. [6] proved that *approximating the AIM problem to within any constant factor is NP-hard.*

Existing solution and its limitation. Chen et al. [6] proposed the *Sampled Double Greedy* (SDG) algorithm for AIM-r, where AIM-r is a restricted version of the AIM problem assuming that the rank of the matrix M is r. For $0 < \delta, \epsilon < 1$, SDG returns a $(1 - 1/e - \epsilon)^3$ solution with probability $1 - \delta$. It runs in $O(|V|^r (\log |C|)^r (b_c + b_v)(|C| + |V| + |M| + |E|)\epsilon^{-r-2} \log(1/\delta))$ where $|M|$ is the number of non-zero entries of M. The running time of SDG grows exponentially with the rank of M. Thus, it cannot handle the AIM problem in large social networks (with large $|V|$) unless the rank of M is one. However, the bipartite-graph modeled by M could take any form (e.g., who subscribes what TV programs, who is a member of which community) and it is impractical to make any assumption about the rank of M. Thus, the applicability of SDG is extremely limited.

Motivated by the limitation of SDG, we present a tunable algorithm AIM-α that returns a solution together with a data-dependent approximation ratio $(1 - 1/e - \epsilon) \cdot \rho$ where $\rho \geq \alpha/b_c$. We also present an efficient heuristic algorithm AIM-$\emptyset$ for the AIM problem.

Both AIM-α and AIM-$\emptyset$ could be easily adapted to handle the AIM problem under a more general model where the probability that a user $v \in V$ being influenced by seed content providers is defined by a more general function $f_v(X, Y)$ that is submodular on both X and Y. For ease of presentation, in this paper, we focus on the amphibious propagation model defined at the beginning of this section (i.e., $f_v(X, Y) = (1 - \prod_{c \in X}(1 - M_{cv})) \cdot \mathbb{I}(v \in Y), \forall v \in V$).

4 A BASIC ALGORITHM

In this section, we present our basic algorithm to AIM. It returns a *basic solution* with $\alpha \leq b_c$ seed content providers and b_v seed users. With high probability, the expected influence spread of the *basic solution* is at least $(1 - 1/e - \epsilon) \cdot \alpha/b_c \cdot OPT$, where OPT is the optimal influence spread. The basic algorithm together with a post-optimization step constitutes our solution AIM-α.

In what follows, we first present how we estimate the influence spread given seeds. Then, we describe how we select seed users given seed content providers. Finally, we elaborate in details about how we find the basic solution.

4.1 Influence estimation

The basic algorithm, AIM-α and AIM-$\emptyset$ all use a FindUser subroutine to find seed users with given content providers $X \subseteq C$. Before

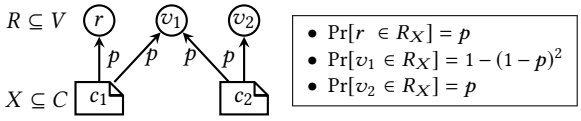


Figure 2: Generating a random RR$_X$-set given a random RR-set R={r, v_1, v_2} and seed content providers X={c_1, c_2}.

describing FindUser, we first present how we estimate the influence spread of $\sigma(X, Y)$ for all $Y \subseteq V$, when $X \subseteq C$ is given.

In the remaining of this paper, we still use "RR-set" to refer to RR-sets for estimating influence spread in the classical influence propagation model (e.g., IC or LT). And the notation R still denotes a random RR-set. Given a set of content providers X, we use "RR$_X$-set" to denote the customized RR-set-like data structure for estimating $\sigma(X, Y)$ for $Y \subseteq V$. Let R_X denote a random RR$_X$-set. Formally, a random RR$_X$-set R_X is defined as follows.

Definition 4.1 (Random RR$_X$-set). Given seed content providers $X \subseteq C$, a random RR$_X$-set R_X is generated as follows. First, we generate a random RR-set R as if we are solving the influence maximization problem. To be specific, we first uniformly and random select a root node $r \in V$, generate a random reverse influence propagation process, and keep all nodes that can influence r in the process in a node set R. Then, for every node $v \in R$, we keep it in R_X with probability $1 - \prod_{c \in X}(1 - M_{cv})$ (i.e., the probability that it can be influenced by X).

Note that if every user can be influenced by the given seed content provider set with probability one, the definition of the random RR$_X$-set is equivalent to that of the random RR-set.

Figure 2 illustrates how we generate a random RR$_X$-set R_X given the content provider set $X = \{c_1, c_2\}$. First, we obtain a random RR-set $R = \{r, v_1, v_2\}$. In Figure 2, every content provider in X influences every user in R with probability p. To get R_X from R, we keep r (resp. v_1 or v_2) in R_X with probability p (resp. $1-(1-p)^2$ or p). Note that although R is always non-empty because $r \in R$, it is possible that none of the nodes in R is included in R_X and $R_X = \emptyset$.

By Definition 4.1, we have the following lemma about RR$_X$-sets.

Lemma 4.2 (RR$_X$-sets). *Given a set X of seed content providers, for any seed user set Y, we have $\sigma(X, Y) = n \cdot \mathbb{E}_{R_X}[\mathbb{I}(R_X \cap Y \neq \emptyset)]$ where he expectation is taken over the randomness of R_X.*

Proof. In the IM problem, let R be a random RR-set, the probability that $R \cap S \neq \emptyset$ equals the probability that a random user can be influenced by the seed set S. Similarly, in the amphibious influence propagation model, let $R_X \subseteq V$ be a random RR$_X$-set where $X \subseteq C$ is a set of seed content providers. Let $Y \subseteq V$ be a set of seed users. The probability that $R_X \cap Y \neq \emptyset$ equals the probability that a random user can be influenced by seed users in Y who are influenced by X. Thus, $\mathbb{E}_{R_X}[\mathbb{I}(R_X \cap Y \neq \emptyset)]$ equals $\sigma(X, Y)/n$. □

4.2 Subroutine for finding seed users

We now present an important FindUser subroutine. Given a set of content providers X, FindUser aims at finding a set Y of b_v seed users so that $\sigma(X, Y)$ is maximized. Algorithm 1 depicts FindUser.

In Algorithm 1, we first compute the probability that X can influence every node v, denoted by $p_X[v]$. The values of $p_X[\cdot]$ are useful

Algorithm 1: FindUser (given content providers X)

Input : $X \subseteq C$, b_v, ϵ,δ, a sequence $\mathcal{R}$ of random RR-sets
Output: (X, Y), $\hat{\sigma}(X, Y)$, an updated sequence $\mathcal{R}$ of RR-sets
// Initialize the probability that X can influence each user
1 **for** $v \in V$ **do** $p_X[v] \leftarrow 1 - \prod_{c \in X}(1 - M_{cv})$
// IM where random RR$_X$-sets are generated via RRGivenX
2 $\langle Y, \hat{\sigma}, \mathcal{R} \rangle \leftarrow$ InfluenceMaximization$(b_v, \epsilon, \delta, \mathcal{R})$ // e.g., SSA
3 **return** (X, Y), $\hat{\sigma}(X, Y) := \hat{\sigma}, \mathcal{R}$
4 **Procedure** RRGivenX
5 $R \leftarrow$ the next RR-set from $\mathcal{R}$ (insert one if there is none)
6 $R_X \leftarrow$ every node $v \in R$ is included in R_X w.p. $p_X[v]$
7 **return** R_X // R_X is a random subset of R

in sampling RR$_X$-sets. Line 2 utilizes an influence maximization algorithm (e.g., SSA) to sample RR$_X$-sets and greedily select a set of b_v seed users according to their estimated influence spread.

Reusing random RR-sets. FindUser takes a sequence of random RR-sets $\mathcal{R}$ as one input and returns it. The returned sequence may contain more random RR-sets. In our algorithm, we reuse the same sequence $\mathcal{R}$ of random RR-sets among all calls to FindUser and other similar subroutines requiring random RR-sets. This is motivated by the fact that sampling a random RR$_X$-set given $X \subseteq C$ requires a random RR-set, but sampling a random RR-set is done independently of X. Thus, we reuse previously generated RR-sets to speed up later calls to FindUser and similar subroutines. Because RR-sets in $\mathcal{R}$ are independent of each other, in each call to FindUser given X, the set of random RR$_X$-sets are independent of each other.

Lemma 4.3 shows the performance guarantee of FindUser.

Lemma 4.3 (FindUser). *Given $X \subseteq C$, b_v, ϵ, and δ. Let Y be the set of seed users returned by* FindUser*, and let Y^* be the optimal solution. With a probability of at least $1-\delta$, Y is a $(1-1/e-\epsilon)$-approximate solution, and we have $(1+\epsilon_a) \cdot \sigma(X, Y) \geq \hat{\sigma}(X, Y)$ and $\hat{\sigma}(X, Y^*) \geq (1-\epsilon_b) \cdot \sigma(X, Y^*)$ where $(1-1/e)(\epsilon_a+\epsilon_b)/(1+\epsilon_a) \leq \epsilon$.*

Proof. By the definition of random RR$_X$-sets, we have $\sigma(X, Y) = n \cdot \mathbb{E}_{R_X}[\mathbb{I}(R_X \cap Y \neq \emptyset)]$ for all $Y \subseteq V$. FindUser samples random RR$_X$-sets to estimate the influence spread. Then, Lemma 4.3 follows from Theorem 2.1 about influence maximization algorithms. □

In Lemma 4.3, the values of ϵ_a and ϵ_b depend on the IM algorithm that being used. We do not consider D-SSA because the values of ϵ_a and ϵ_b are determined during the runtime and may vary from different calls to FindUser. Both IMM and SSA could be applied in FindUser, we choose SSA for its efficiency: it is more efficient than IMM while being used as a building block in our algorithms. While presenting our algorithms, we assume that ϵ_a and ϵ_b are known because our analysis is independent of their specific values. We will provide detailed parameter settings for our experiments later.

4.3 Design of the basic algorithm

The basic algorithm finds α seed content providers and b_v seed users. The naive approach is to enumerate over all $\binom{n_c}{\alpha}$ combinations of α content providers. For each combination X, we find seed users Y and get a candidate solution (X, Y). Finally, we return the candidate solution with the largest estimated influence spread.

Algorithm 2: Basic Algorithm

Input : $B, G, \alpha, b_v, \epsilon, \delta$
Output: (X_α, Y_α), $\hat{\sigma}(X_\alpha, Y_\alpha)$, a sequence of random RR-sets $\mathcal{R}$

// Preparation
1 $\mathcal{R} \leftarrow \emptyset$, $(X_\alpha, Y_\alpha) \leftarrow (\emptyset, \emptyset)$, $\hat{\sigma}(X_\alpha, Y_\alpha) \leftarrow 0$, $prune_{thld} \leftarrow 0$
2 **for** $c \in C$ **do** $ub[c] \leftarrow$ `InfluenceUB`$(\{c\})$
3 Sort content providers so that $ub[c_i] \geq ub[c_{i+1}], \forall i$
// Enumeration
4 **for** *each* $X \subseteq C$, $|X| = \alpha$, *in dictionary order* **do**
5 $check \leftarrow$`PruneCheck`(X)
6 **if** *check is "do not prune"* **then**
7 $\langle (X, Y), \hat{\sigma}(X, Y), \mathcal{R} \rangle \leftarrow$ `FindUser`$(X, b_v, \epsilon, \delta, \mathcal{R})$
8 **if** $\hat{\sigma}(X, Y) \geq \hat{\sigma}(X_\alpha, Y_\alpha)$ **then**
9 $(X_\alpha, Y_\alpha) \leftarrow (X, Y)$, $\hat{\sigma}(X_\alpha, Y_\alpha) \leftarrow \hat{\sigma}(X, Y)$
10 $prune_{thld} \leftarrow \hat{\sigma}(X_\alpha, Y_\alpha)/(1 + \epsilon_a)$
11 **else if** *check is "early termination"* **then** break
12 **return** (X_α, Y_α), $\hat{\sigma}(X_\alpha, Y_\alpha)$, $\mathcal{R}$
// Return an upper bound of $\sigma(X, V)$ with prob. at least $1 - \delta$
13 **Procedure** `InfluenceUB`(X)
14 $\mathcal{R}_X \leftarrow \theta$ random RR$_X$-sets use random RR-sets in $\mathcal{R}$ (add more random RR-sets to $\mathcal{R}$ if necessary)
15 $\hat{\sigma}(X, V) \leftarrow \frac{n}{\theta} \sum_{R_X \in \mathcal{R}_X} \mathbb{I}(R_X \cap V \neq \emptyset)$
16 **return** $\hat{\sigma}(X, V) + n \cdot \sqrt{\frac{1}{2\theta} \cdot \ln(\frac{1}{\delta})}$
17 **Procedure** `PruneCheck`(X)
18 **if** $\alpha = 1$ **then**
19 $ub_X \leftarrow ub[c]$, where c in the only element in X
20 **if** $ub_X < prune_{thld}$ **then return** *"early termination"*
21 **else**
22 $ub_X \leftarrow$ `InfluenceUB`(X)
23 **if** $ub_X < prune_{thld}$ **then return** *"prune X"*
24 **return** *"do not prune"*

Our basic algorithm is inspired by the above idea, but works in a smarter way. We are able to prune "hopeless" content provider sets without having to find seed users for them by being careful about the "enumeration order" of content provider sets. Algorithm 2 depicts the basic algorithm.

Preparation. Line 1 initializes the sequence of random RR-sets $\mathcal{R}$, the basic solution and the "pruning-threshold". The pruning-threshold maintains a lower bound of the influence spread of the basic solution. Lines 2-3 sort content providers in ascending order of their upper bounds of influence spread when being used as a seed returned by the `InflueneUB` subroutine. For a set $X \subseteq C$, $\sigma(X, V)$ is an upper bound of the influence spread of X and any set of seed users $Y \subseteq V$. Because of the #P-hardness of computing $\sigma(X, V)$, we obtain an upper bound of $\sigma(X, V)$ with high probability via a subroutine `InfluenceUB`. We will present more details about the `InfluenceUB` subroutine and the pruning strategy shortly.

Enumeration. Lines 4-11 enumerate over all α-combinations of content providers in the dictionary order of $ub[\cdot]$. For example, when $\alpha = 2$ and $|C| \geq 2$, the first two enumerated sets are $\{c_0, c_1\}$ and $\{c_0, c_2\}$. For $X \subseteq C$, `PruneCheck` decides whether we can skip finding seed users given X or even terminate the enumeration. We find seed users given X only if the estimated upper bound of $\sigma(X, V)$ exceeds the pruning-threshold. Suppose X is not pruned, Lines 7-10 find seed users for it and maintain the basic solution and the pruning-threshold. Finally, Line 12 returns the best solution.

`InfluenceUB`. Given X, `InfluenceUB` estimates $\sigma(X, V)$ via θ samples of RR$_X$-sets. Then, it derives the additive error of the estimation by Chernoff bound, and returns an upper bound of $\sigma(X, V)$. For the value of θ, we find that $\theta = n/10$ is a balanced choice: it is usually no more than the number of RR-sets required in all calls to `FindUser`, and it also provides a tight upper bound so that the pruning strategy is effective. Formally, we have the following lemma about `InfluenceUB`. The proof is presented in the appendix.

Lemma 4.4 (`InfluenceUB`). *Given $X \subseteq C$, `InfluenceUB` returns an upper bound of $\sigma(X, V)$ with a probability of at least $1 - \delta$.*

Pruning. We prune subsets $X \subseteq C$ that are unlikely to be returned as seed content providers. Recall that (X_α, Y_α) keeps the best solution found so far. If we know $\sigma(X, V) \leq \sigma(X_\alpha, Y_\alpha)$ for $X \subseteq C$, we can prune X because we know we have $\sigma(X, Y) \leq \sigma(X_\alpha, Y_\alpha)$ for any set Y of seed users. Because of the #P-hardness of the influence computation, to check whether we can prune $X \subseteq C$, `PruneCheck` compares an upper bound of $\sigma(X, V)$ and a lower bound of $\sigma(X_\alpha, Y_\alpha)$ maintained by the pruning-threshold. We prune a set X if the upper bound of $\sigma(X, V)$ is no larger than the pruning-threshold.

Analysis. Theorem 4.5 shows the influence spread of the basic solution. The $(1 - 1/e - \epsilon) \cdot \alpha/b_c$-approximation guarantee of our algorithm `AIM-`α that will be presented in the next section is mainly due to Theorem 4.5.

Theorem 4.5 (Basic algorithm). *Let (X^*_α, Y^*_α) be the optimal solution when we can select α seed content providers and b_v seed customers. And let $OPT_\alpha = \sigma(X^*_\alpha, Y^*_\alpha)$. Denote the basic solution returned by the basic algorithm by (X_α, Y_α). With a probability of at least $1 - \binom{n_c}{\alpha} + 1) \cdot \delta$, we have $\hat{\sigma}(X_\alpha, Y_\alpha) \geq (1-1/e)(1-\epsilon_b) \cdot OPT_\alpha$, $\hat{\sigma}(X_\alpha, Y_\alpha) \leq (1+\epsilon_a) \cdot \sigma(X_\alpha, Y_\alpha)$ where $(1-1/e)(\epsilon_a+\epsilon_b)/(1+\epsilon_a) \leq \epsilon$. Thus, with a probability of at least $1 - \binom{n_c}{\alpha} + 1) \cdot \delta$, we have*

$$\sigma(X_\alpha, Y_\alpha) \geq (1 - 1/e - \epsilon) \cdot OPT_\alpha. \tag{3}$$

Proof Outline. From Lemma 4.3 and Lemma 4.4, we can show that the following two arguments hold simultaneously with a high probability. First, we do not prune the set X^*_α. We obtain a set Y' of seed users for X^*_α, and $\hat{\sigma}(X^*_\alpha, Y') \geq (1-1/e)(1-\epsilon_b) \cdot OPT_\alpha$. Second, the basic solution (X_α, Y_α) satisfies $\hat{\sigma}(X_\alpha, Y_\alpha) \leq (1+\epsilon_a) \cdot \sigma(X_\alpha, Y_\alpha)$. When the above two arguments hold, and because $(1 - 1/e)(\epsilon_a + \epsilon_b)/(1 + \epsilon_a) \leq \epsilon$, we could show by contradiction that $\sigma(X_\alpha, Y_\alpha) \geq (1 - 1/e - \epsilon) \cdot OPT_\alpha$. The complete proof is in the appendix. □

5 APPROXIMATION ALGORITHM

We now present our algorithm `AIM-`α to the AIM problem. `AIM-`α is based on the basic algorithm and an extra post optimization phase. In this section, we first present an important subroutine `FindContentProvider` required by the post optimization phase. Then, we present `AIM-`α by elaborating the post optimization phase in detail and providing analysis of the approximation ratio.

Algorithm 3: `FindContentProvider` (given users Y)

Input : $Y \subseteq V$, b_c, a sequence of random RR-sets $\mathcal{R}$
Output: (X, Y)

1 $\mathcal{R}_Y \leftarrow |\mathcal{R}|$ random RR$_Y$-sets via `RRGivenY`$(B, G, \mathcal{R})$
2 Find $X \subseteq C$ with b_c nodes that "covers" most RR$_Y$-sets in $\mathcal{R}_Y$
3 **return** (X, Y)
4 **Procedure** `RRGivenY`$(B, G, \mathcal{R})$
5 $R \leftarrow$ the next unused RR-set from $\mathcal{R}$, $R_Y \leftarrow \emptyset$
6 **for** $c \in R \cap Y$ **do** Insert c into R_Y w.p. $1 - \prod_{v \in R \cap Y}(1 - M_{cv})$
7 **return** R_Y // R_Y is a random subset of $R \cap Y$

5.1 Subroutine for finding seed content providers

`FindContentProvider` finds seed content providers given a set of seed users Y. It is similar to `FindUser`. First, we define the random RR$_Y$-set that will be used to estimate the influence spread $\sigma(X, Y)$ for $X \subseteq C$ when the set of seed user Y is given.

Definition 5.1 (A random RR$_Y$-set R_Y). Given a set of seed users $Y \subseteq V$, a random RR$_Y$-set R_Y is generated as follows. First, we generate a random RR-set R as if we are solving the influence maximization problem. Then, we insert every $c \in C$ connected to nodes in $R \cap Y$ into R_Y with the probability that c can influence at least one node in $R \cap Y$ (i.e., with probability $1 - \prod_{v \in R \cap Y}(1 - M_{cv})$).

We have the following lemma about RR$_Y$-sets.

Lemma 5.2 (RR$_Y$-sets). *Given $Y \subseteq V$, for any seed content provider set $X \subseteq C$, we have $\sigma(X, Y) = n \cdot \mathbb{E}_{R_Y}[\mathbb{I}(R_Y \cap X \neq \emptyset)]$. The expectation is taken over the randomness of the RR$_Y$-set R_Y.*

Proof. Let R_Y be a random RR$_Y$-set ($R_Y \subseteq C$). Given $X \subseteq C$, the probability that $R_Y \cap X \neq \emptyset$ equals the probability that a random user is influenced by seed users in Y who are influenced by X. □

Algorithm 3 depicts `FindContentProvider`. First, we construct a set $\mathcal{R}_Y$ of random RR$_Y$-sets. Then, we greedily select and return a set of b_c content providers that "covers" the most RR$_Y$-sets in $\mathcal{R}_Y$: we say that $X \subseteq C$ "covers" an RR$_Y$-set R_Y iff. $X \cap R_Y \neq \emptyset$. In practice, the number of RR-sets in $\mathcal{R}$ generated in the basic algorithm is not too small. Because $|\mathcal{R}_Y| = |\mathcal{R}|$, the number of random RR$_Y$-sets in $\mathcal{R}_Y$ is not too small. By applying the Chernoff bound, we know that $n/|\mathcal{R}_Y| \cdot \sum_{R_Y \in \mathcal{R}_Y} \mathbb{I}(R_Y \cap X \neq \emptyset)$ well approximates $\sigma(X, Y)$ for all $X \subseteq C$. In other words, the number of RR$_Y$-sets covered by $X \subseteq C$ is proportional to $\sigma(X, Y)$. Thus, if the greedy algorithm selects a subset $X \subseteq C$ of seed content providers given seed users Y, it is very likely that $\sigma(X, Y)$ is large.

5.2 Algorithm design of `AIM-`α

Now, we are ready to present our algorithm `AIM-`α. Algorithm 4 depicts `AIM-`α. We divide Algorithm 4 into two phases.

(1) *Basic phase (Line 2).* Via Algorithm 2, we obtain a basic solution and a sequence of random RR-sets generated for later usages.
(2) *Post-optimization (Lines 3-16).* Lines 3-11 find three candidate solutions. Then, Lines 12-15 return the best one. In case we select the basic solution as the final solution, which rarely happens in practice, Line 15 inserts nodes $c \in C$ into X_α in descending order of $ub[c]$ until X_α has b_c nodes. Here, $ub[c]$ is an upper bound of $\sigma(\{c\}, V)$ computed in the basic algorithm.

Algorithm 4: `AIM-`α

Input : $B, G, b_c, b_v, \epsilon, \delta$
Output: (X, Y), $\hat{\sigma}(X, Y)$, approx. ratio $(1 - 1/e - \epsilon) \cdot \rho$

1 $\delta' \leftarrow \delta / \left(\binom{n_c}{\alpha} + 4\right)$
// Utilize the basic algorithm in Algorithm 2
2 $\langle (X_\alpha, Y_\alpha), \hat{\sigma}(X_\alpha, Y_\alpha), \mathcal{R} \rangle \leftarrow$ `BasicAlgo`$(B, G, \alpha, b_v, \epsilon, \delta')$
// Refining the basic solution
3 $(X_1, Y_1) \leftarrow$`FindContentProvider`$(Y_\alpha, b_c, \mathcal{R})$
4 $\langle (X_1, Y_1), \hat{\sigma}(X_1, Y_1), \mathcal{R} \rangle \leftarrow$ `FindUser`$(X_1, b_v, \epsilon, \delta', \mathcal{R})$
// Assembling top seed content provider subsets
5 **for** X *not pruned in the basic algo., in desc. order of* $\hat{\sigma}(X, Y)$ **do**
6 Insert $c \in X$ into X_2 in desc. order of $ub[c]$ until $|X_2| = b_c$
7 Insert $c \in C$ into X_2 in desc. order of $ub[c]$ until $|X_2| = b_c$
8 $\langle (X_2, Y_2), \hat{\sigma}(X_2, Y_2), \mathcal{R} \rangle \leftarrow$ `FindUser`$(X_2, b_v, \epsilon, \delta', \mathcal{R})$
// Finding seed users first
9 $\left\langle (C, Y_3'), \hat{\sigma}(C, Y_3), \mathcal{R} \right\rangle \leftarrow$ `FindUser`$(C, b_v, \epsilon, \delta', \mathcal{R})$
10 $(X_3, Y_3') \leftarrow$`FindContentProvider`$(Y_3', b_c, \mathcal{R})$
11 $\langle (X_3, Y_3), \hat{\sigma}(X_3, Y_3), \mathcal{R} \rangle \leftarrow$ `FindUser`$(X_3, b_v, \epsilon, \delta', \mathcal{R})$
// Return the best solution
12 $i^* \leftarrow \arg\max_{i \in \{\alpha, 1, 2, 3\}} \hat{\sigma}(X_i, Y_i)$
13 $\rho \leftarrow \frac{\hat{\sigma}(X_{i^*}, Y_{i^*})}{\hat{\sigma}(X_\alpha, Y_\alpha)} \cdot \frac{\alpha}{b_c}$
14 **if** $|X| < b_c$ **then** // happens only if $i^* = \alpha$
15 Insert $c \in C$ into X in desc. order of $ub[c]$ until $|X| = b_c$
16 **return** (X_{i^*}, Y_{i^*}), $\hat{\sigma}(X_{i^*}, Y_{i^*})$, $(1 - 1/e - \epsilon) \cdot \rho$

Post-optimization in details. We now describe how we obtain three candidate solutions and the intuitions behind.

Refining the basic solution (Lines 3-4). Given the basic solution (X_α, Y_α), we first find a seed content provider set X_1 given Y_α via `FindContentProvider`. Then, we find a seed user set Y_1 given X_1 via `FindUser`. If both `FindUser` and `FindContentProvider` return optimal solutions, we expect to have $\sigma(X_1, Y_1) \geq \sigma(X_1, Y_\alpha) \geq \sigma(X_\alpha, Y_\alpha)$.

Assembling top seed content provider subsets (Lines 5-8). At a high level, we first fill up X_2 with sized-α subsets of C with high influence spread if seed users are chosen properly. In case X_2 contains less than b_c elements, we insert $c \in C$ into X_2 in a descending order of $ub[c]$ until $|X_2| = b_c$. Given $X_2 \subseteq C$, we find seed users Y_2 via `FindUser`. If `FindUser` returns optimal solutions, we expect to have $\sigma(X_2, Y_2) \geq \sigma(X_2, Y_\alpha) \geq \sigma(X_\alpha, Y_\alpha)$ because $X_\alpha \subseteq X_2$.

Finding seed users first (Lines 9-11). First, we find a set of seed users Y_3' via `FindUser` assuming we are seeding all content providers. Then, we find a set of seed content providers X_3 given Y_3', and finalize a set of seed users Y_3 given X_3. This candidate solution would have an influence spread larger than $\sigma(X_\alpha, Y_\alpha)$, if we can find a subset X_3 so that $\sigma(X_3, Y_3') \geq \sigma(X_\alpha, Y_\alpha)$, and then we have $\sigma(X_3, Y_3) \geq \sigma(X_3, Y_3') \geq \sigma(X_\alpha, Y_\alpha)$.

Performance analysis. Theorem 5.3 shows the influence spread of the solution returned by Algorithm 4.

Algorithm 5: AIM-$\emptyset$

Input : $B, G, b_c, b_v, \epsilon, \delta$
Output: (X, Y), estimated influence $\hat{\sigma}(X, Y)$

Initialize $\mathcal{R}$ as an empty sequence of random RR-sets
// Assembling top seed content provider (adapted)
Let $ub[c] \leftarrow$ InfluenceUB($\{c\}$), $\forall c \in C$
Insert $c \in C$ into X_1 in desc. order of $ub[c]$ until $|X_1| = b_c$
$\langle (X_1, Y_1), \hat{\sigma}(X_1, Y_1), \mathcal{R} \rangle \leftarrow$ FindUser($X_1, b_v, \epsilon, \delta, \mathcal{R}$)
// Finding seed users first
$\langle (C, Y_2), \hat{\sigma}(C, Y_2), \mathcal{R} \rangle \leftarrow$ FindUser($C, b_v, \epsilon, \delta, \mathcal{R}$)
$(X_2, Y_2) \leftarrow$FindContentProvider($Y_2, b_c, \mathcal{R}$)
$\langle (X_2, Y_2), \hat{\sigma}(X_2, Y_2), \mathcal{R} \rangle \leftarrow$ FindUser($X_2, b_v, \epsilon, \delta, \mathcal{R}$)
// Return the better solution
$i^* \leftarrow \arg\max_{i \in \{1,2\}} \hat{\sigma}(X_i, Y_i)$
return (X_{i^*}, Y_{i^*}), $\hat{\sigma}(X_{i^*}, Y_{i^*})$

Theorem 5.3 (AIM-α). *Suppose* AIM-α *returns a solution* (X, Y) *and a data-dependent approximation ratio* $(1 - 1/e - \epsilon) \cdot \rho$. *We have* $\rho \geq \alpha / b_c$. *Moreover, with a probability of at least* $1 - \delta$, *we have*

$$\sigma(X, Y) \geq (1 - 1/e - \epsilon) \cdot \rho \cdot OPT. \tag{4}$$

Proof. First, because $\hat{\sigma}(X_{i^*}, Y_{i^*}) \geq \hat{\sigma}(X_\alpha, Y_\alpha)$, we have $\rho \geq \alpha / b_c$. Next, we prove that Inequality (4) holds with probability at least $1 - \delta$. Suppose we have $\hat{\sigma}(X_\alpha, Y_\alpha) \geq (1 - 1/e)(1 - \epsilon_b) \cdot OPT_\alpha$, which happens with probability $1 - \left(\binom{n_c}{\alpha} + 1\right)\delta'$ according to Theorem 4.5 about the basic algorithm. Furthermore, suppose we have $\hat{\sigma}(X_i, Y_i) \leq (1 + \epsilon_a) \cdot \sigma(X_i, Y_i)$ for all $1 \leq i \leq 3$, which happens with a probability of at least $1 - 3\delta'$ from Lemma 4.3. Let (X_{i^*}, Y_{i^*}) be the final solution, we have $\sigma(X, Y) = \sigma(X_{i^*}, Y_{i^*}) \geq \frac{\hat{\sigma}(X_{i^*}, Y_{i^*})}{\hat{\sigma}(X_\alpha, Y_\alpha)} \cdot \frac{\hat{\sigma}(X_\alpha, Y_\alpha)}{1+\epsilon_a} \geq \frac{\hat{\sigma}(X_{i^*}, Y_{i^*})}{\hat{\sigma}(X_\alpha, Y_\alpha)} \cdot \frac{(1-1/e)(1-\epsilon_b)}{1+\epsilon_a} \cdot OPT_\alpha \geq (1 - 1/e - \epsilon) \cdot \rho \cdot OPT$. The last inequality holds because we have $OPT_\alpha \geq \frac{\alpha}{b_c} \cdot OPT$ from the submodularity of $\sigma(X, Y)$ on X, and we have $(1 - 1/e)(1 - \epsilon_b)/(1 + \epsilon_a) \geq (1 - 1/e - \epsilon)$ from $(1 - 1/e)(\epsilon_a + \epsilon_b)/(1 + \epsilon_a) \leq \epsilon$. By union bound, Inequality (4) holds with probability at least $1 - \left(\binom{n_c}{\alpha} + 4\right)\delta' = 1 - \delta$. □

6 HEURISTIC ALGORITHM

In this section, we present a heuristic solution AIM-$\emptyset$ to the AIM problem. In AIM-α, the enumeration step in searching the basic solution is fundamental to the approximation guarantee and the data-dependent approximation ratio. However, the enumeration step is the most time-consuming part of AIM-α in our experiments. This is because we have to find seed users for different α-combinations of the content provider set C. In practice, we observe that we are able to identify solutions with large influence spread without the enumeration step. This motivates us to design a heuristic algorithm AIM-$\emptyset$ that sacrifices the performance guarantee in exchange for the greater efficiency.

Algorithm 5 depicts the AIM-$\emptyset$ algorithm. It removes the enumeration step from the basic algorithm of AIM-α and adapts the way we find candidate solutions. In Algorithm 5, Lines 1-4 find the first candidate solution (X_1, Y_1). We let set X_1 be the set of nodes with top-b_c values of the estimated upper bound $ub[\cdot]$, and we find a set of seed users Y_1 given X_1 via FindUser. Lines 5-7 find the second candidate solution (X_2, Y_2) in the same way as we obtain the third candidate solution in Algorithm 4. Finally, the solution with larger estimated influence spread is returned. In AIM-$\emptyset$, parameters ϵ and δ control the balance between the efficiency and effectiveness of FindUser, FindContentProvider and thus the entire algorithm. We found in our experiments that $\epsilon = 0.5$ and $\delta = 0.001$ provide a good balanced efficiency and effectiveness.

Due to the removal of the basic algorithm, AIM-$\emptyset$ does not provide approximation guarantee. Nevertheless, our experiments show that AIM-$\emptyset$ returns solution with similar influence spread as compared to AIM-α. Moreover, AIM-$\emptyset$ is much more efficient than AIM-α. Thus, when the performance guarantee is not required, AIM-$\emptyset$ is an efficient alternative to AIM-α.

7 EXPERIMENTS

We conduct extensive experiments to test the performance of our proposed algorithms AIM-α and AIM-$\emptyset$.

Table 2: Summaries of datasets

Dataset	Social network G		Bipartite graph B		
	$\|V\|$	$\|E\|$	$\|C\|$	$\|M\|$	Type
Enron-Email [15]	37K	368K	20	50K	synthetic
Slashdot [18]	77K	828K	20	50K	synthetic
BrightKite [9]	58K	428K	49	22K	check-in
Gowalla [9]	197K	1.9M	49	47K	check-in
YouTube [31]	1.1M	6.0M	16K	129K	community
Flickr [20]	1.7M	22.6M	104K	8.5M	community

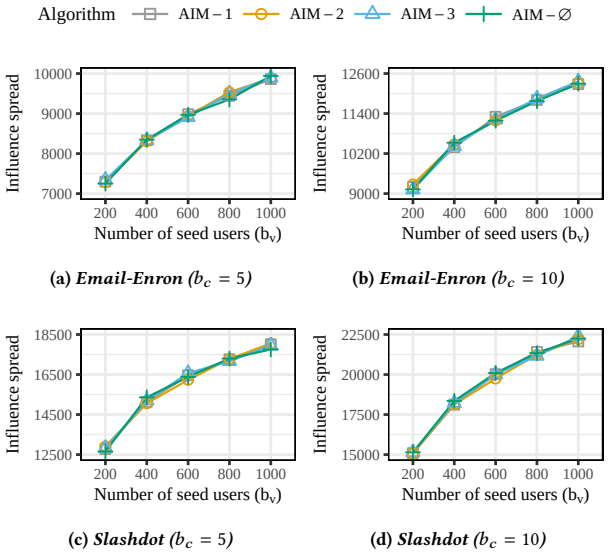
Datasets. Table 2 summaries the datasets. We categorized them into two classes by how we obtain the bipartite graph.

Partially synthetic datasets: *Enron-Email* is an email communication network. *Slashdot* is a social network. We generate a small bipartite graph synthetically. To be specific, we randomly generate 50,000 edges between users and 20 content providers.

Datasets with known community structures: (1) *BrightKite* and *Gowalla* are social networks where users share check-ins to locations. In both datasets, more than half of the check-ins happened in the US. We construct a bipartite graph where each state or Washington, D.C. in mainland US is represented by a content provider node c. There is an edge between a node c and a user v if v checked-in to the location represented by c. (2) *YouTube* and *Flickr* are social networks with ground-truth communities. We use the community affiliation graph as the bipartite graph: a node $c \in C$ connects to a user v if and only if v is a member of the community c.

In social networks, we assign influence probabilities according to the *Weighted Cascade* model [14]. For each edge $e_{uv} \in E$, we let $p_{uv} = 1/d_v^-$ where d_v^- is the indegree of v. In bipartite graphs, each edge e_{cv} is assigned with a random probability $M_{cv} \sim U[0, 1]$. For each dataset, we conduct experiments under both IC and LT model. Due to space limit, most results where influence propagates under the LT model are omitted and included in our technical report [1].

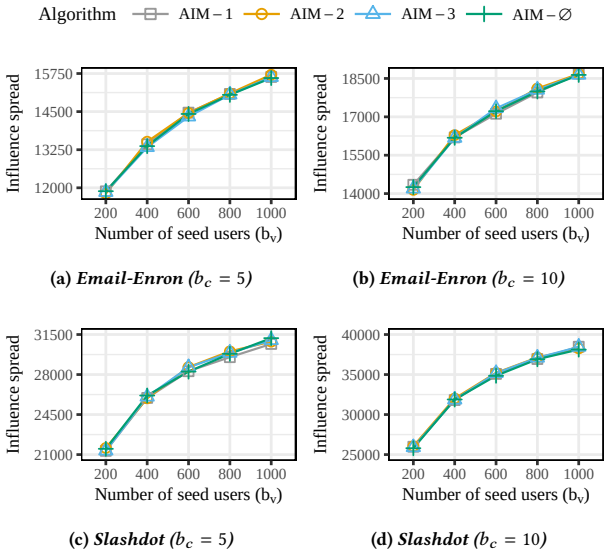
Other settings. In FindUser and FindContentProvider, we use the SSA method to find seed users or seed content providers. We let $\epsilon_a = (1 + \epsilon/4)(1 + \epsilon/3) - 1$ and $\epsilon_b = \epsilon/(3(1 - 1/e))$ for SSA. For both



(a) *Email-Enron* ($b_c = 5$) (b) *Email-Enron* ($b_c = 10$)

(c) *Slashdot* ($b_c = 5$) (d) *Slashdot* ($b_c = 10$)

Figure 3: Influence spread on partially synthetic datasets under the IC model.



(a) *Email-Enron* ($b_c = 5$) (b) *Email-Enron* ($b_c = 10$)

(c) *Slashdot* ($b_c = 5$) (d) *Slashdot* ($b_c = 10$)

Figure 4: Influence spread on partially synthetic datasets under the LT model.

AIM-α and AIM-$\emptyset$, we let $\epsilon = 0.5$ and $\delta = 0.001$. Thus, AIM-α returns a $(1 - 1/e - \epsilon) \cdot \alpha/b_c$-approximate solution with probability at least 99.9%. We re-evaluate the influence spread of the final solution to within a relative error of 1%. We repeat each experiment ten times and report the averaged results. All experiments were conducted on a Linux machine with an Intel Xeon E5-1620v2@3.70GHz CPU and 16GB memory.

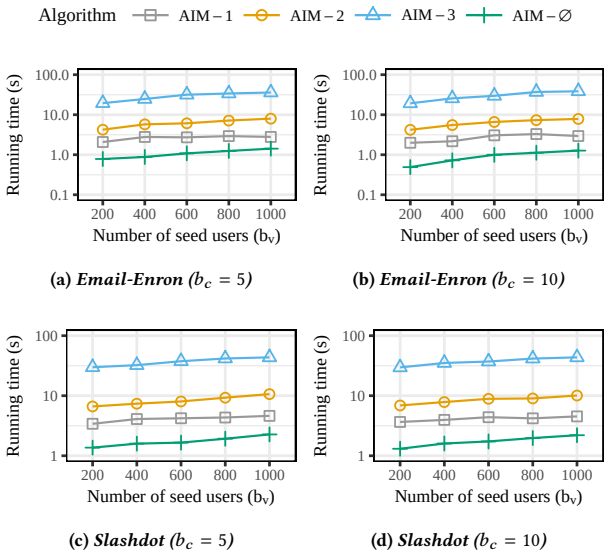
We do not compare our algorithms with the SDG algorithm [6] because it *cannot run for any of our dataset*: Recall that SDG [6] runs in $O(|V|^r(\log|C|)^r(b_c + b_v)(|C| + |V| + |M| + |E|)\epsilon^{-r-2}\log(1/\delta))$, where r is the rank of M and $|M|$ is the number of non-zero entries of M. In all of our dataset, the rank of the bipartite graph adjacency matrix M is at least 20, thus SDG cannot accomplish the task.

7.1 Partially synthetic datasets

In this subsection, we report results on datasets with synthetic bipartite graphs. We examine the performance of AIM-α with different values of α and the AIM-$\emptyset$ algorithm. The algorithm AIM-α where $\alpha = i$ is denoted by AIM-i (e.g., AIM-1).

Influence spread. Figure 3 and Figure 4 show the influence spread of solutions returned by AIM-1, AIM-2, AIM-3, and AIM-$\emptyset$ where the influence propagates in the social network under the IC model and the LT model. We observe that all four algorithms return solutions with comparable influence spread. This implies that although the approximation guarantee provided by AIM-1 is smaller than those provided by AIM-α where $\alpha > 1$, AIM-1 is in practice as effective as AIM-α where $\alpha > 1$. The observation that AIM-$\emptyset$ returns solution with high influence spread implies that the two candidate solutions in AIM-$\emptyset$ adapted from the post-optimization step of AIM-α are very effective.

Running time. Figure 5 and Figure 6 show the running time of AIM-1, AIM-2, AIM-3, and AIM-$\emptyset$ where the influence propagates in the social network under the IC model and the LT model. Given the two-layer network, b_c and b_v, the running time of AIM-α and AIM-$\emptyset$ is proportional to the number of calls to the FindUser subroutine. Algorithm AIM-$\emptyset$ calls the FindUser subroutine $O(1)$ times. Algorithm AIM-α calls the FindUser subroutine $O(\binom{n_c}{\alpha})$ times. Accordingly, Figure 5 and Figure 6 show that AIM-$\emptyset$ runs faster than



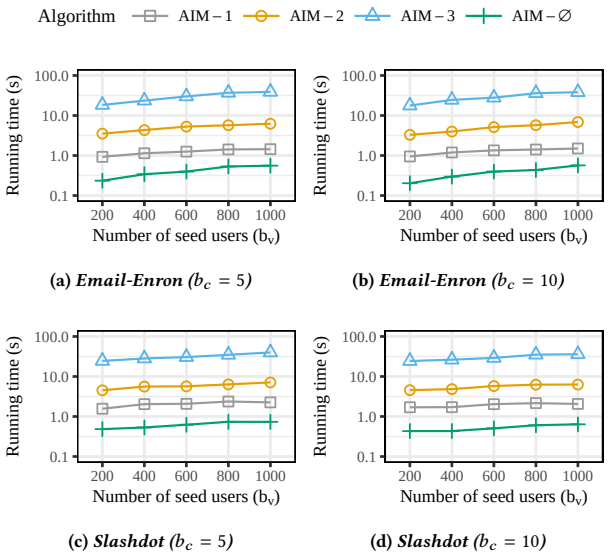
(a) *Email-Enron* ($b_c = 5$) (b) *Email-Enron* ($b_c = 10$)

(c) *Slashdot* ($b_c = 5$) (d) *Slashdot* ($b_c = 10$)

Figure 5: Running time on partially synthetic datasets under the IC model (log scale).

Table 3: Comparing the approximation ratio $(1-1/e-\epsilon)\cdot\rho$ and $(1-1/e-\epsilon)\cdot\alpha/b_c$ on partially synthetic datasets.

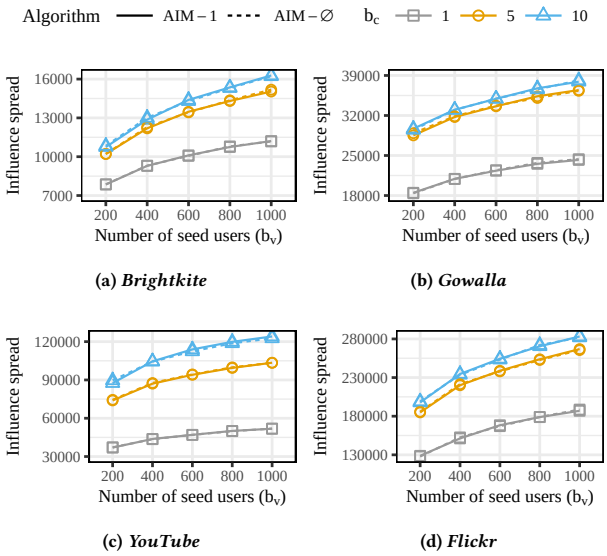
Ratio	Dataset (Model)	$b_c=5, b_v=200$			$b_c=5, b_v=1000$			$b_c=10, b_v=200$			$b_c=10, b_v=1000$		
		$\alpha=1$	$\alpha=2$	$\alpha=3$	$\alpha=1$	$\alpha=2$	$\alpha=3$	$\alpha=1$	$\alpha=2$	$\alpha=3$	$\alpha=1$	$\alpha=2$	$\alpha=3$
ρ	*Enron-Email* (IC)	0.38	0.60	0.73	0.38	0.58	0.73	0.23	0.38	0.45	0.23	0.38	0.45
	Enron-Email (LT)	0.38	0.56	0.70	0.38	0.54	0.69	0.23	0.36	0.45	0.23	0.30	0.42
	Slashdot (IC)	0.38	0.53	0.68	0.45	0.61	0.74	0.23	0.30	0.38	0.26	0.38	0.45
	Slashdot (LT)	0.42	0.58	0.70	0.45	0.61	0.75	0.23	0.37	0.45	0.30	0.38	0.45
α/b_c	all datasets	0.20	0.40	0.60	0.20	0.40	0.60	0.10	0.20	0.30	0.10	0.20	0.30



(a) *Email-Enron* ($b_c=5$) (b) *Email-Enron* ($b_c=10$)

(c) *Slashdot* ($b_c=5$) (d) *Slashdot* ($b_c=10$)

Figure 6: Running time on partially synthetic datasets under the LT model (log scale).



(a) *Brightkite* (b) *Gowalla*

(c) *YouTube* (d) *Flickr*

Figure 7: Influence spread on datasets with known communities under the IC model.

AIM-α for all tested α, and the running time of AIM-α grows exponentially with α. Moreover, Figure 5 and Figure 6 show that the running time increases when the number of seed users b_v increases. This is because FindUser takes more time to find more seed users when b_v increases. To conclude, AIM-$\emptyset$ is a good choice to the AIM problem if performance guarantee is not needed. Otherwise, AIM-1 is an efficient solution.

Approximation ratio of AIM-α. AIM-α returns a solution (X, Y) together with an approximation ratio $(1-1/e-\epsilon)\cdot\rho$ where $\rho \geq \alpha/b_c$. We have $\sigma(X,Y) \geq (1-1/e-\epsilon)\cdot\rho\cdot OPT$ with high probability. Table 3 compares the values of ρ and α/b_c. We observe that ρ is up to three times as large as α/b_c. Recall that Algorithm 4 computes ρ as follows: $\rho = \frac{\hat{\sigma}(X_{i^*},Y_{i^*})}{\hat{\sigma}(X_\alpha,Y_\alpha)}\cdot\frac{\alpha}{b_c}$. When $\alpha < b_c$, the value of ρ is larger than α/b_c meaning that the first fraction is always greater than one, which implies that the post-optimization step always finds a final solution with a larger influence spread as compared with the basic solution. From Table 3, we can conclude that we can always obtain an approximation guarantee better than $(1-1/e-\epsilon)\cdot\alpha/b_c$.

7.2 Datasets with known communities

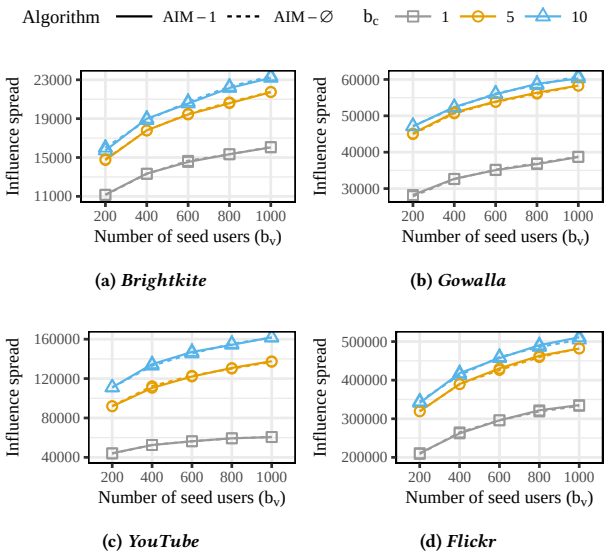
Now, we report results on datasets where the bipartite graphs are constructed according to check-ins of users or ground-truth communities. To avoid cluttering the results, for AIM-α, we only report performance of AIM-1.

Influence spread. Figure 7 and Figure 8 show the influence spread of solution returned by AIM-1 and AIM-$\emptyset$, with varying number of seed content providers b_c and seed users b_v. We observe that AIM-1 and AIM-$\emptyset$ always return solutions with similar influence spread. Thus, both AIM-1 and AIM-$\emptyset$ are very effective.

Pruning strategy. Table 4 shows the number and the fraction of subsets $X \subseteq C$ *not pruned* by the basic algorithm AIM-1. Note that the fraction of subsets not pruned is the number of such subsets over $\binom{n_c}{\alpha} = n_c$ ($\alpha = 1$). Table 4 shows that more than 88.37% of subsets are pruned for *BrightKite* and *Gowalla*, and more than 99.70% subsets are pruned for *YouTube* and *Flickr*. Thus, our pruning strategy in AIM-1 is very effective, and we only have to find seed users for a very small fraction of subsets. The reason behind the effectiveness of pruning is as follows. We observe from our datasets that only a few content providers connect to many users; and most

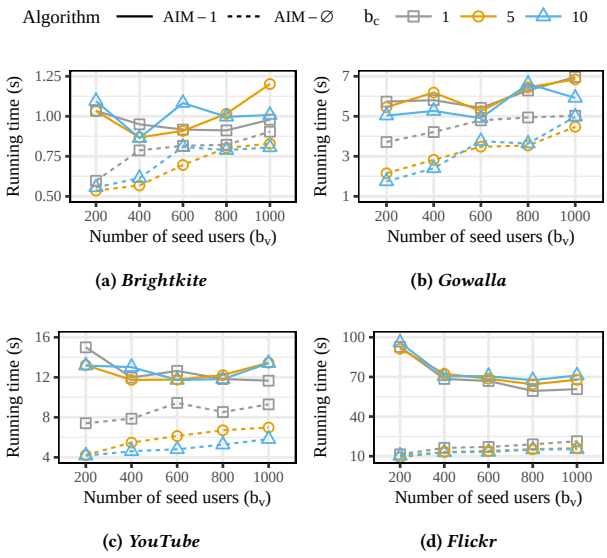
Table 4: The averaged number and fraction of subsets $X \subseteq C$ for which we find seed users (i.e., not pruned) in the basic algorithm of AIM-1.

Dataset (Model)	$b_c = 1$			$b_c = 5$			$b_c = 10$		
	$b_v = 200$	$b_v = 600$	$b_v = 1000$	$b_v = 200$	$b_v = 600$	$b_v = 1000$	$b_v = 200$	$b_v = 600$	$b_v = 1000$
BrightKite (IC)	5.90 (12.04%)	3.00 (6.12%)	3.00 (6.12%)	5.70 (11.63%)	3.00 (6.12%)	2.50 (5.10%)	5.90 (12.04%)	3.00 (6.12%)	2.90 (5.92%)
BrightKite (LT)	3.90 (7.96%)	3.00 (6.12%)	1.30 (2.65%)	4.00 (8.16%)	2.70 (5.51%)	1.30 (2.65%)	4.30 (8.78%)	2.80 (5.71%)	1.40 (2.86%)
Gowalla (IC)	4.40 (8.98%)	2.80 (5.71%)	2.10 (4.29%)	4.30 (8.78%)	2.80 (5.71%)	2.30 (4.69%)	4.60 (9.39%)	2.80 (5.71%)	2.20 (4.49%)
Gowalla (LT)	3.40 (6.94%)	2.20 (4.49%)	2.00 (4.08%)	3.40 (6.94%)	2.20 (4.49%)	2.00 (4.08%)	3.40 (6.94%)	2.10 (4.29%)	2.00 (4.08%)
YouTube (IC)	6.90 (0.04%)	3.00 (0.02%)	3.00 (0.02%)	6.70 (0.04%)	3.20 (0.02%)	3.00 (0.02%)	6.60 (0.04%)	3.00 (0.02%)	3.00 (0.02%)
YouTube (LT)	5.60 (0.03%)	3.00 (0.02%)	3.00 (0.02%)	5.50 (0.03%)	3.00 (0.02%)	3.00 (0.02%)	5.50 (0.03%)	3.00 (0.02%)	3.00 (0.02%)
Flickr (IC)	315.00 (0.30%)	119.00 (0.11%)	75.10 (0.07%)	308.20 (0.30%)	119.40 (0.12%)	72.30 (0.07%)	312.40 (0.30%)	122.40 (0.12%)	74.40 (0.07%)
Flickr (LT)	244.40 (0.24%)	84.90 (0.08%)	51.90 (0.05%)	243.50 (0.23%)	85.60 (0.08%)	52.20 (0.05%)	244.00 (0.24%)	84.00 (0.08%)	50.50 (0.05%)



(a) ***Brightkite*** (b) ***Gowalla*** (c) ***YouTube*** (d) ***Flickr***

Figure 8: Influence spread on datasets with known communities under the LT model.



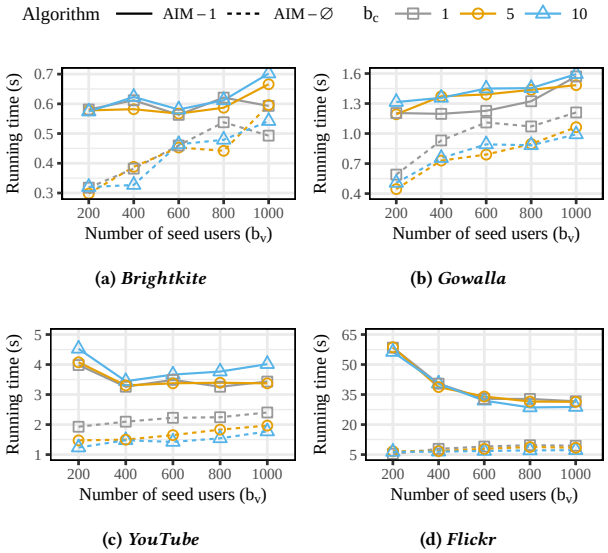
(a) ***Brightkite*** (b) ***Gowalla*** (c) ***YouTube*** (d) ***Flickr***

Figure 9: Running time on datasets with known communities under the IC model.

content providers connect to a relatively smaller number of users. In *BrightKite*, California has 454K check-ins, Texas has 216K check-ins, only five states have more than 100K check-ins, and the averaged number of check-ins is only about 51K. In *Gowalla*, Texas has 790K check-ins, California has 571K check-ins, only six states have more than 100K check-ins, and the averaged number of check-ins is only around 64K. In *YouTube* and *Flickr*, "content providers" corresponds to communities; there are some very large communities with large amounts of users, but most communities have a relatively smaller size. In AIM-1, under the presence of content providers connecting to a large number of users, content providers with small degrees are not likely to appear in either the basic solution or the final solution. The pruning strategy of Algorithm 2 is able to prune these content providers with small degrees, and thus a majority of content providers could be pruned.

Computational costs. Figure 9 and Figure 10 show the running time of AIM-1 and AIM-∅ where influence propagates in social networks under the IC model and the LT model. Both of our algorithms are efficient. Moreover, AIM-∅ is more efficient due to the removal of the enumeration part, which plays a fundamental role in providing the performance guarantee of AIM-1. Again, the running time of AIM-1 is proportional to the number of calls to the FindUser. For *BrightKite*, *Gowalla* and *YouTube*, the gaps between the running time of AIM-1 and AIM-∅ are not very large, especially when b_v is large. This is because the number of calls to FindUser is only a few (less than ten), as shown in Table 4. For *Flickr*, the gaps between the running time of AIM-1 and AIM-∅ are relatively larger than other datasets. This is because the number of calls to FindUser is tens or hundreds. Moreover, for each b_c, we observe that the gaps diminish when b_v increases in all four datasets. This is mainly because the number of calls to the FindUser subroutine decreases when b_v increases, as shown in Table 4.

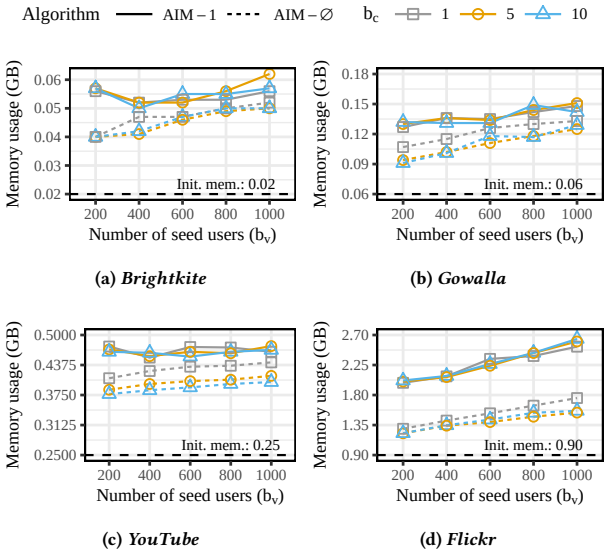
Figure 11 and Figure 12 show the memory usage of AIM-α and AIM-∅. We also label the "intial memory" of our algorithms. This is the memory usage of our algorithm right after we have loaded the datasets into memory. Experimental results show that the memory usage of both our algorithms are at most three times as large as the memory needed to loading the dataset. Moreover, AIM-1 consumes a



(a) *Brightkite* (b) *Gowalla*

(c) *YouTube* (d) *Flickr*

Figure 10: Running time on datasets with known communities under the LT model.



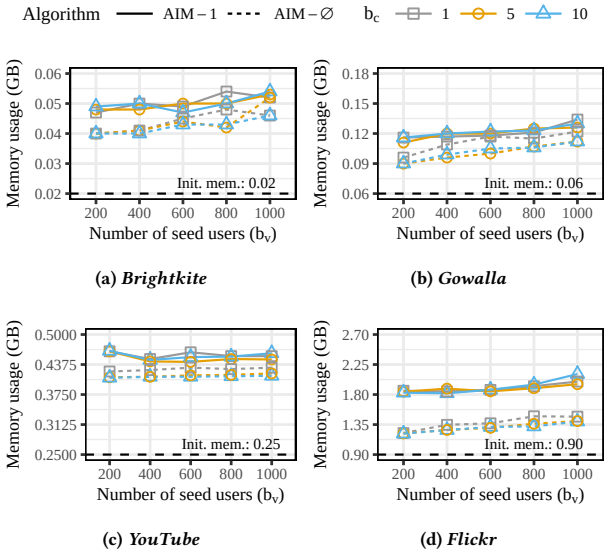
(a) *Brightkite* (b) *Gowalla*

(c) *YouTube* (d) *Flickr*

Figure 11: Memory usage on datasets with known communities under the IC model.

bit more memory as compared to AIM-∅. This is reasonable because AIM-1 may require more random RR-sets (or, RR_X-sets, RR_Y-sets) to provide the approximation guarantee.

Approximation ratio. Table 5 show the approximation guarantee of AIM-1. The data-dependent approximation ratio $(1-1/e-\epsilon)\cdot\rho$ is always larger than $(1-1/e-\epsilon)\cdot\alpha/b_c$. For $b_c=5$ and $b_c=10$, ρ (resp. $(1-1/e-\epsilon)\cdot\rho$) is up to two times as large as α/b_c (resp.



(a) *Brightkite* (b) *Gowalla*

(c) *YouTube* (d) *Flickr*

Figure 12: Memory usage on datasets with known communities under the LT model.

Table 5: Approximation ratios $(1-1/e-\epsilon)\cdot\rho$ and $(1-1/e-\epsilon)\cdot\alpha/b_c$ on datasets with known community structures.

Ratio	Dataset (Model)	b_c=5		b_c=10	
		b_v=200	b_v=1000	b_v=200	b_v=1000
ρ	*BrightKite* (IC)	0.26	0.30	0.15	0.15
	BrightKite (LT)	0.29	0.30	0.15	0.15
	Gowalla (IC)	0.30	0.30	0.15	0.15
	Gowalla (LT)	0.32	0.30	0.15	0.15
	YouTube (IC)	0.39	0.39	0.23	0.23
	YouTube (LT)	0.40	0.45	0.23	0.29
	Flickr (IC)	0.30	0.30	0.15	0.15
	Flickr (LT)	0.30	0.30	0.15	0.15
α/b_c	all datasets	0.20	0.20	0.10	0.10

$(1-1/e-\epsilon)\cdot\alpha/b_c$). For $b_c=1$, we have $\rho=\alpha/b_c=1$ in AIM-1. From Table 5 and Table 3, we can conclude that AIM-α returns solutions with a data-dependent approximation guarantee larger than $(1-1/e-\epsilon)\cdot\alpha/b_c$ when $\alpha<b_c$.

8 RELATED WORK

We have summarized works related to the classical influence maximization problems in Section 2. In this section, we summarize two lines of studies related to the AIM problem [6] we study in this paper. A series of works studied the "adaptive seeding" problem [3, 12, 16, 24, 25, 28]. In these problems, the seeding process has multiple stages. Initially, only some budget is used to target influencers. The remaining budget will be spent after observing the influence diffusion. These "adaptive seeding" problems are similar to AIM in that they both consider a multi-stage influence maximization. However, in AIM, we simultaneously select both seed

content providers and seed consumers. And, unlike the adaptive seeding problem, the AIM problem is inapproximable. Another line of related work studied how to increase acceptance probabilities of important users [19, 30] or how to inject links to social networks via friend recommendation [2, 5, 22, 23] so to boost the influence spread. In these works, boosted users or injected links help to increase the influence spread. Similarly, in AIM, seed users help to spread the influence of seed content providers to other users in the social network. The main difference is that seed content providers cannot influence non-seed users directly in AIM.

9 CONCLUSION

In this work, we design efficient algorithms for solving the *Amphibious Influence Maximization* (AIM) problem. Given a bipartite graph $B = (C, V, M)$ and a social network $G = (V, E)$, AIM asks to select b_c seed content providers $X \subseteq C$ and b_v seed users $Y \subseteq V$ so that the expected influence spread $\sigma(X, Y)$ is maximized.

We develop AIM-α that returns $(1-1/e-\epsilon)\cdot\alpha/b_c$-approximate solution with high probability. Together with a solution (X, Y), AIM-α also returns a data-dependent approximation ratio $(1 - 1/e - \epsilon) \cdot \rho$ indicating that (X, Y) is a $(1-1/e-\epsilon)\cdot\rho$-approximate solution with high probability. We also present an efficient heuristic algorithm AIM-$\emptyset$. We conduct extensive experiments on datasets with real social networks and bipartite graphs constructed in various ways. Experimental results demonstrate the superiority of AIM-α (especially AIM-1) and AIM-$\emptyset$. In particular, we show that AIM-1 returns solution with similar influence spread as compared to AIM-α with $\alpha > 1$. Thus, if performance guarantee is needed, AIM-1 is an efficient choice. For the heuristic algorithm AIM-$\emptyset$, it is more efficient than AIM-1, and it returns solution with influence spread similar to the solution returned by AIM-α in all our experiments. Thus, if one is indifferent about the performance guarantee, AIM-$\emptyset$ is a good choice to the AIM problem. Moreover, if we have a dynamic two-layer network, we could update seeds via the *much more efficient* AIM-$\emptyset$ algorithm, and it is very likely that the updated seeds have similar influence spread as compared to what would be returned by the relatively less efficient algorithm AIM-α.

REFERENCES

[1] Technical report: Where & who should you advertise? influence maximization for two-layer networks. https://www.dropbox.com/s/jtkihbtpmkzolcl/TwoLayerIMTechReport.pdf?dl=0, 2017.
[2] S. Antaris, D. Rafailidis, and A. Nanopoulos. Link injection for boosting information spread in social networks. *SNAM*, 4(1), 2014.
[3] A. Badanidiyuru, C. Papadimitriou, A. Rubinstein, L. Seeman, and Y. Singer. Locally adaptive optimization: Adaptive seeding for monotone submodular functions. In *SODA*, 2016.
[4] C. Borgs, M. Brautbar, J. Chayes, and B. Lucier. Maximizing social influence in nearly optimal time. In *SODA*, 2014.
[5] V. Chaoji, S. Ranu, R. Rastogi, and R. Bhatt. Recommendations to boost content spread in social networks. In *WWW*, 2012.
[6] W. Chen, F. Li, T. Lin, and A. Rubinstein. Combining traditional marketing and viral marketing with amphibious influence maximization. In *EC*, 2015.
[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. Yuan, and L. Zhang. Scalable influence maximization in social networks under the linear threshold model. In *ICDM*, 2010.
[9] E. Cho, S. A. Myers, and J. Leskovec. Friendship and mobility: user movement in location-based social networks. In *KDD*, 2011.
[10] E. Cohen, D. Delling, T. Pajor, and R. F. Werneck. Sketch-based influence maximization and computation: Scaling up with guarantees. In *CIKM*, 2014.
[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] T. Horel and Y. Singer. Scalable methods for adaptively seeding a social network. In *WWW*, 2015.
[13] K. Huang, S. Wang, G. S. Bevilacqua, X. Xiao, and L. V. S. Lakshmanan. Revisiting the stop-and-stare algorithms for influence maximization. *PVLDB*, 10(9):913–924, 2017.
[14] D. Kempe, J. Kleinberg, and É. Tardos. Maximizing the spread of influence through a social network. In *KDD*, 2003.
[15] B. Klimt and Y. Yang. Introducing the enron corpus. In *CEAS*, 2004.
[16] S. Lattanzi and Y. Singer. The power of random neighbors in social networks. In *WSDM*, 2015.
[17] J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. VanBriesen, and N. Glance. Cost-effective outbreak detection in networks. In *ICDM*, 2007.
[18] J. Leskovec, K. J. Lang, A. Dasgupta, and M. W. Mahoney. Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters. *Internet Mathematics*, 6(1), 2009.
[19] Y. Lin, W. Chen, and J. C. S. Lui. Boosting information spread: An algorithmic approach. In *ICDE*, 2017.
[20] A. Mislove, M. Marcon, K. P. Gummadi, P. Druschel, and B. Bhattacharjee. Measurement and analysis of online social networks. In *SIGCOMM*, 2007.
[21] H. T. Nguyen, M. T. Thai, and T. N. Dinh. Stop-and-stare: Optimal sampling algorithms for viral marketing in billion-scale networks (exteneded version of the same sigmod'16 conference paper). *CoRR*, abs/1605.07990v3, 2016.
[22] D. Rafailidis and A. Nanopoulos. Crossing the boundaries of communities via limited link injection for information diffusion in social networks. In *WWW Companion*, 2015.
[23] D. Rafailidis, A. Nanopoulos, and E. Constantinou. "with a little help from new friends". *Journal of System and Software*, 98(C), 2014.
[24] A. Rubinstein, L. Seeman, and Y. Singer. Approximability of adaptive seeding under knapsack constraints. In *EC*, 2015.
[25] L. Seeman and Y. Singer. Adaptive seeding in social networks. In *FOCS*, 2013.
[26] Y. Tang, X. Xiao, and Y. Shi. Influence maximization: Near-optimal time complexity meets practical efficiency. In *SIGMOD*, 2014.
[27] Y. Tang, X. Xiao, and Y. Shi. Influence maximization in near-linear time: A martingale approach. In *SIGMOD*, 2015.
[28] G. Tong, W. Wu, S. Tang, and D.-Z. Du. Adaptive influence maximization in dynamic social networks. *IEEE/ACM Transactions on Networking*, PP(99), 2016.
[29] X. Wang, Y. Zhang, W. Zhang, X. Lin, and C. Chen. Bring order into the samples: A novel scalable method for influence maximization. *TKDE*, 2016.
[30] 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.
[31] J. Yang and J. Leskovec. Defining and evaluating network communities based on ground-truth. *Knowledge and Information Systems*, 42(1), 2015.

APPENDIX

Proof of Lemma 4.4. By Chernoff bound, if S is the sum of θ independent $\{0, 1\}$ variables with mean μ, we have $\Pr[S/\theta \le \mu - \epsilon] \le \exp(-2\theta\epsilon^2)$ for all $0 < \epsilon < 1$. Let $sum_X = \hat{\sigma}(X, V) \cdot \theta/n$. By definition of the RR_X-set, sum_X is the sum of θ independent $\{0, 1\}$ variables with the mean of $\sigma(X, V)/n$. By Chernoff bound, we have

$$\Pr\left[n \cdot \left(sum_X/\theta + \sqrt{\ln(1/\delta)/(2\theta)}\right) < \sigma(X, V)\right]$$
$$= \Pr\left[sum_X/\theta \le \sigma(X, V)/n - \sqrt{\ln(1/\delta)/(2\theta)}\right] \le \delta.$$

Thus, the returned value is an upper bound of $\sigma(X, V)$ with a probability of at least $1 - \delta$. □

Proof of Theorem 4.5. We first define two critical events.

- **Event 1**: In each call to FindUser in Algorithm 2 given $X \subseteq C$, let Y be the returned set of seed users, both of the following two inequalities hold (inequalities in Lemma 4.3),

$$(1 + \epsilon_a) \cdot \sigma(X, Y) \ge \hat{\sigma}(X, Y), \tag{5}$$
$$\hat{\sigma}(X, Y^*) \ge (1 - \epsilon_b) \cdot \sigma(X, Y^*), \tag{6}$$

  where $(1 - 1/e)(\epsilon_a + \epsilon_b)/(1 + \epsilon_a) \le \epsilon$.
- **Event 2**: If $\alpha = 1$, we have $ub[c^*] \ge \sigma(X^*_\alpha, V)$ where c^* is the only element in X^*_α. If $\alpha > 1$, while checking whether we can prune X^*_α, the estimated upper bound ub_X in Line 22 of Algorithm 2 satisfies $ub_X \ge \sigma(X^*_\alpha, V)$.

From Lemma 4.3, the first event happens with proabaiblity at least $1-\binom{n_c}{\alpha}\cdot\delta$. From Lemma 4.4, the second event happens with proabaiblity at least $1-\delta$. With probability $1-\left(\binom{n_c}{\alpha}+1\right)$, both the above events happen. Now, suppose both events happen. At all times during the execution of Algorithm 2, the pruning-threshold is assigned to be $\hat{\sigma}(X_\alpha, Y_\alpha)/(1+\epsilon_a)$ which is as most $\sigma(X_\alpha, Y_\alpha)$ according to Inequality (5). Therefore, the pruning-threshold never exceeds $\sigma(X^*_\alpha, V)$, and we know that X^*_α must not be pruned.

Let Y' be the set of users we select for X^*_α, we have

$$\begin{aligned}
&\hat{\sigma}(X_\alpha, Y_\alpha) \geq \hat{\sigma}(X^*_\alpha, Y') \\
\geq&(1-1/e)\cdot \max_{Y\subseteq V, |Y|=b_v} \hat{\sigma}(X^*_\alpha, Y) && \text{(submodularity of } \hat{\sigma}) \\
=&(1-1/e)\cdot \hat{\sigma}(X^*_\alpha, Y^*_\alpha) && \text{(optimality of } Y^*) \\
\geq&(1-1/e)(1-\epsilon_b)\cdot \sigma(X^*_\alpha, Y^*_\alpha) && \text{(Inequality (6))} \\
=&(1-1/e)(1-\epsilon_b)\cdot OPT_\alpha
\end{aligned}$$

We show by contradiction that $\sigma(X_\alpha, Y_\alpha) \geq (1-1/e-\epsilon)\cdot OPT_\alpha$. Assume $\sigma(X_\alpha, Y_\alpha) < (1-1/e-\epsilon)\cdot OPT_\alpha$, we have

$$\begin{aligned}
&\sigma(X_\alpha, Y_\alpha) \geq \hat{\sigma}(X_\alpha, Y_\alpha) - \epsilon_a\cdot\sigma(X_\alpha, Y_\alpha) \quad \text{(Inequality (5))} \\
&\geq (1-1/e)(1-\epsilon_b)\cdot OPT_\alpha - \epsilon_a\cdot(1-1/e-\epsilon)\cdot OPT_\alpha \\
&\geq \left\{(1-1/e)\left(1-\left(\frac{\epsilon(1+\epsilon_a)}{1-1/e}-\epsilon_a\right)\right)-\epsilon_a(1-1/e-\epsilon)\right\}\cdot OPT_\alpha \\
&= (1-1/e-\epsilon)\cdot OPT_\alpha,
\end{aligned}$$

which contradicts with the assumption. □