Online Influence Maximization

Siyu Lei
University of Hong Kong
Pokfulam Road, Hong Kong
sylei@cs.hku.hk

Silviu Maniu
Noah's Ark Lab, Huawei
Science Park, Hong Kong
silviu.maniu@huawei.com

Luyi Mo
University of Hong Kong
Pokfulam Road, Hong Kong
lymo@cs.hku.hk

Reynold Cheng
University of Hong Kong
Pokfulam Road, Hong Kong
ckcheng@cs.hku.hk

Pierre Senellart
Télécom ParisTech; CNRS LTCI
& NUS; CNRS IPAL
pierre@senellart.com

ABSTRACT

Social networks are commonly used for marketing purposes. For example, free samples of a product can be given to a few influential social network users (or *seed nodes*), with the hope that they will convince their friends to buy it. One way to formalize this objective is through the problem of *influence maximization* (or IM), whose goal is to find the best seed nodes to activate under a fixed budget, so that the number of people who get influenced in the end is maximized. Solutions to IM rely on the *influence probability* that a user influences another one. However, this probability information may be unavailable or incomplete.

In this paper, we study IM in the absence of complete information on influence probability. We call this problem *Online Influence Maximization* (OIM), since we learn influence probabilities at the same time we run influence campaigns. To solve OIM, we propose a multiple-trial approach, where (1) some seed nodes are selected based on existing influence information; (2) an influence campaign is started with these seed nodes; and (3) user feedback is used to update influence information. We adopt Explore–Exploit strategies, which can select seed nodes using either the current influence probability estimation (exploit), or the confidence bound on the estimation (explore). Any existing IM algorithm can be used in this framework. We also develop an incremental algorithm that can significantly reduce the overhead of handling user feedback information. Our experiments show that our solution is more effective than traditional IM methods on the partial information.

1. INTRODUCTION

In recent years, there has been a lot of interest about how social network users can affect or *influence* others (via the so-called *word-of-mouth* effect). This phenomenon has been found to be useful for marketing purposes. For example, many companies have advertised their products or brands on social networks by launching influence campaigns, giving free products to a few influential individuals (seed nodes), with the hope that they can promote the products to their friends [20]. The objective is to identify a set of most influential people, in order to attain the best marketing effect. This problem of *influence maximization* (IM) has attracted a lot of research interest [6, 7, 9, 10, 23].

KDD'15, August 10-13, 2015, Sydney, NSW, Australia.

DOI: http://dx.doi.org/10.1145/2783258.2783271.

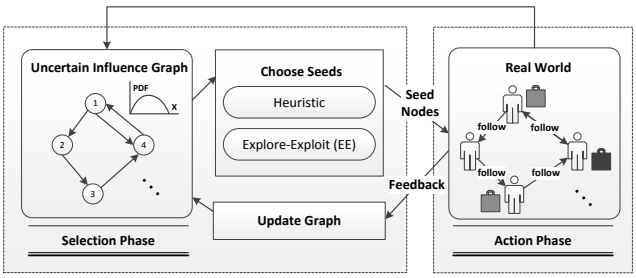


Figure 1: The OIM framework.

Given a *promotion budget*, the goal of IM is to select the best seed nodes from an *influence graph*. An influence graph is essentially a graph with *influence probabilities* among nodes representing social network users. In the *independent cascade model*, for example, a graph edge e from user a to b with influence probability p implies that a has a chance p to affect the behavior of b (e.g., a convinces b to buy a movie ticket) [16]. Given an influence graph, IM aims to find k *seed nodes*, whose expected number of influenced nodes, or *influence spread*, is maximized. Marketing efforts can then be focused on the k nodes (or persons). In the IM literature, these seed nodes are said to be *activated* [6, 7, 9, 10, 23].

While existing IM algorithms effectively obtain the most influential seed nodes, they assume that the influence probability value between each pair of nodes is known. However, this assumption may not hold. Consider a marketing firm starting in a new city with some knowledge of the social network of the users in the city. The company, however, does not know how influence propagates among these users. Unless the influence probability information is known, the marketing firm cannot run an IM algorithm and decide the target users. To obtain these values, *action logs*, which record the social network user's past activities, can be used [11]. This information may not be readily available.

Is it possible to perform IM on a social network, even if the information about influence probabilities is *absent* or *incomplete*? We call this problem *Online Influence Maximization* (OIM), as we aim at discovering influence probabilities at the same time we are performing influence campaigns. (We say that an IM algorithm is *offline*, if it assumes that the influence probability between every node pair is known in advance.) In the absence of complete influence probability information, making the best marketing effort out of a limited promotion budget can be challenging. To tackle this problem, we propose a solution based on influencing seed nodes in *multiple*

rounds. In each round, we select some seed nodes to activate (e.g., advertising a product to a few selected users). The feedback of these users is then used to decide the seed nodes to be activated in the next round. The information about influence probabilities in the social network is learnt and refined during these campaigns.

Figure 1 illustrates our OIM framework. It contains multiple successive influence campaigns, or *trials*. A trial should fulfill one of two objectives: (1) to advertise to promising nodes; and (2) to improve the knowledge about influence probabilities. A trial consists of two phases: *selection* and *action*. In the *selection phase*, an *uncertain influence graph* is maintained. This graph models the uncertainty of influence probabilities among social network users, in terms of a probability distribution. A *seed selection strategy*, based on an existing IM solution, is then executed on this graph to produce up to k seed nodes. In the *action phase*, the selected seed nodes are activated in the *real world* (e.g., sending the advertisement message to chosen users). The actions of these users, or *feedback* (e.g., whether the message is further spread), is then used to *update* the uncertain influence graph. The iteration goes on, until the marketing budget is exhausted. In this paper, we focus on the two crucial components of the selection phase: (1) *seed selection strategy*; and (2) techniques for *updating* the uncertain influence graph.

1. Seed selection strategy. To choose seed nodes in a trial, a simple way is to make use of existing IM algorithms. Due to the lack of knowledge about influence probabilities, this approach may not be the best. We thus develop an *Explore–Exploit strategy* (or *EE*), which performs IM based on existing influence probability information:

- **[Exploit]** Select k seed nodes for getting the most rewarding influence spread from the influence graph, derived from the uncertain influence graph. Any state-of-the-art IM algorithms (e.g., CELF [19], DD [7], TIM and TIM+ [27]) can be used; or
- **[Explore]** Select k seed nodes based on some strategy (e.g., through estimating the confidence bound of the influence probability) to improve the knowledge about the influence graph.

In this paper, we study strategies for exploit and explore. With suitable use of strategies, *EE* performs better than running an existing IM algorithm on the uncertain influence graph alone.

In our OIM solution, N trials are carried out. In each trial, an existing IM algorithm may be executed. If N is large, the performance of our algorithm can be affected. The problem is aggravated if the underlying uncertain influence graph is big. For state-of-the-art IM algorithms (e.g., CELF [19] and TIM+ [27]), this running time is dominated by the cost of sampling the influence graph. For example, in TIM+, the sampling effort costs more than 99% of the computation time. We design an efficient solution, based on the intuition that users' feedback often only affects a small portion of the influence graph. If samples of the previous iterations are stored, it is possible to reuse them, instead of sampling the influence graph again. We examine conditions allowing a sampled graph to be effectively reused in a new trial. We propose an *incremental algorithm*, and present related data structures for facilitating efficient evaluation of our solution. This algorithm can support any sample-based IM algorithm running on independent cascade models. We demonstrate how to use TIM+ in this paper.

2. Updating the uncertain influence graph. As discussed before, our seed selection strategy is executed on the uncertain influence graph (Figure 1). It is important that this graph accurately reflects the current knowledge about the influence among different users, so that the seed selection strategy can make the best decision. We investigate algorithms for updating this graph based on the feedback of activated users (e.g., whether they spread out an advertisement message). We examine two variants, which update the influence graph *locally* and *globally*. A local update refreshes the parameters of the influence probability distribution between two graph nodes, while a global update is applied to the parameters of the influence probability information that applies to the whole uncertain influence graph. These algorithms are developed based on classical machine learning methods (e.g, Least Squares Estimation and Maximum Likelihood).

Our solutions can be adopted by social marketers who aim to promote their products, in cases when the underlying probabilities of the influence graph are unknown. Our approach can utilize any state-of-the-art IM algorithm. We also examine how to update the uncertain influence graph effectively by machine learning methods. We develop an incremental algorithm to improve the efficiency of our solution. Our experiments demonstrate that our proposed methods can effectively and efficiently maximize influence spread.

2. RELATED WORK

Influence Maximization (IM). Kempe et al. [16] first proposed the study of IM in social networks. They showed that finding the set of seed nodes that maximizes influence is NP-hard, and showed that the greedy algorithm has a constant approximation guarantee. However, this solution is not very fast, because thousands of samples are often required, and each sampling operation has a complexity linear to the graph size. To improve the efficiency of IM solutions, several heuristics were developed, namely Degree Discount [7], PMIA [6], IPA [17], and IRIE [15]. Although these heuristics are fast, their accuracy is not theoretically guaranteed. Improved aproximation algorithms with theoretical guarantees include CELF [19], CELF++ [13], and NewGreedy [7]. More recently, Borgs et al. proposed an algorithm based on reverse influence sampling, and showed that it is runtime-optimal with accuracy guarantees [4]. The scalability of this solution was enhanced by Tang et al., who developed TIM and TIM+ [27] to further reduce the number of samples needed.

There are also other works that address different variants of the IM problem: (1) incorporating community [28] and topic [2] information in the propagation process; (2) competition of different parties for influence [21]; and (3) use of other influence propagation models such as linear threshold or credit distribution [12, 14, 26].

Learning influence probabilities. Saito et al. [25] modeled the problem of obtaining influence probabilities as an instance of likelihood maximization, and developed an expectation maximization algorithm to solve it. Given a social network and an *action log* (e.g., user u performs action a at time t), Goyal et al. [11] developed static and time-dependent models to compute influence probabilities between a pair of social network users. These methods require the action log information of all the users involved to be known in advance; however, this information may not be available. Our framework does not require all action logs to be available. Instead, we select seed nodes in multiple advertising campaigns, so that influence maximization can be done faster. We then use users' feedback in each campaign to learn and refine influence probabilities.

Multi-armed bandits (MAB). The EE strategy in the seed selection phase of our solution is inspired by the ε-greedy algorithm, which was originally developed to solve the *multi-armed bandit problem* (MAB) [24]. In the ε-greedy algorithm [22], ε controls the trade-off between exploitation and exploration. Specifically, with probability 1 - ε, an action is executed based on the current knowledge (i.e., *exploit*); with probability ε, another action is performed (i.e., *explore*). This framework is adopted as a baseline in our solution.

Table 1: Symbols used in this paper.

symbol	description
G	influence graph
V	set of users (nodes) of G
E	set of edges of G
p_{ij}	influence probability from i to j (fixed value)
P_{ij}	influence probability from i to j (random variable)
N	number of trials
k	budget for each trial
S	set of seed nodes
$\sigma(S)$	expected influence spread
(α,β)	global prior for the beta distribution
A_n	set of successfully activated nodes in trial n
F_n	real world activation feedback in trial n
(h_{ij},m_{ij})	number of successful and unsuccessful activations of the edge from i to j

[8] studies combinatorial MAB algorithms, and in particular the CUCB algorithm, which uses upper confidence bounds [3] for choosing between explore and exploit. A scenario akin to the OIM problem is illustrated and it is shown that CUCB achieves a bound on the regret. However, CUCB is not applicable due to two factors. First, the activated nodes are counted multiple times leading to redundant activations and choices. Second, and most practically important, the approximation bound depends on an initialization step in which each arm (in this scenario, seed node) is tested to get an activation feedback; this is not practically feasible in cases when activation budgets are limited. Another algorithm closely related to our framework is Thompson Sampling [1], where each independent arm is simulated by a Beta distribution of successes and failures. In our scenario, the arms are the parameters of the algorithms, and defining success and failure in a result of an influence maximization is not trivial.

3. INFLUENCE MAXIMIZATION: REVIEW

We now provide a review of the IM problem and its solutions. This forms the basis of the OIM problem to be studied in this paper. Table 1 shows the notation used.

Let $G=(V,E,p)$ be an *influence graph*, where $v\in V$ are *users* or nodes, and $e\in E$ are the links or edges between them. Each edge $e=(i,j)$ between users i and j is associated with an *influence probability* $p_{ij}\in[0,1]$. This value represents the probability that user j is activated by user i at time $t+1$, given that user i is activated at time t. We also suppose that time flows in discrete, equal steps. In the IM literature, p_{ij} is given for every i and j. Obtaining p_{ij} requires the use of action logs [11] which may not be available. In this paper, we investigate how to perform IM without knowing p_{ij} in advance.

In the independent cascade model, at a given timestamp t, every node is in either active (influenced) or inactive state, and the state of each node can be changed from inactive to active, but not vice-versa. When a node i becomes active in step t, the influence is independently propagated at $t+1$ from node i to its currently inactive neighbors with probability p_{ij}. Node i is given one chance to activate its inactive neighbor. The process terminates when no more activations are possible. A node can be independently activated by any of its (active) incoming neighbors. Suppose that the activation process started from a set S of nodes. We call the expected number of activated nodes of S the *expected influence spread*, denoted $\sigma(S)$. Formally:

DEFINITION 1. *Given a weighted graph $G=(V,E,p)$, let* infl *be the immediate influence operator, which is the random process that extends a set of nodes $X\subseteq V$ into a set of immediately influenced nodes* $\mathrm{infl}(X)$*, as follows:*

$$\Pr(v\in\mathrm{infl}(X))=\begin{cases}1 & \text{if } v\in X;\\ 1-\prod_{\substack{(u,v)\in E\\ u\in X}}(1-p_{uv}) & \text{otherwise.}\end{cases}$$

Given a seed set $S\subseteq V$, we define the set of influenced nodes *$I(S)\subseteq V$ as the random variable that is the fixpoint $I^{\infty}(S)$ of the following inflationary random process:*

$$\begin{cases} I^0(S)=\emptyset;\\ I^1(S)=S;\\ I^{n+2}(S)=I^{n+1}(S)\cup\mathrm{infl}(I^{n+1}(S)\backslash I^n(S)) & \text{for } n\geqslant 0.\end{cases}$$

The influence spread $\sigma(S)$ is $\mathbb{E}[|I(S)|]$.

Based on the above definition, [16] defines the *influence maximization problem* (IM) as follows.

PROBLEM 1. *Given a weighted graph $G=(V,E,p)$ and a number $1\leqslant k\leqslant|V|$, the* influence maximization *(IM) problem finds a set $S\subseteq V$ such that $\sigma(S)$ is maximal subject to $|S|=k$.*

As discussed in [16], evaluating the influence spread is difficult. Even when the spread values are known, obtaining an exact solution for the IM problem is computationally intractable. Next we outline the existing IM algorithms for this problem.

IM algorithms. A typical IM algorithm evaluates the *score* of a node based on some metric, and inserts the k *best* nodes, which have the highest scores, into S. For example, the degree discount (DD) heuristic [7] selects the nodes with highest degree as S. Another classical example is *greedy*: at each step, the *next best node*, or the one that provides the largest marginal increase for σ, is inserted into S. This is repeated until $|S|=k$. The *greedy* algorithm provides an $(1-1/e)$-approximate solution for the IM problem. To compute the influence spread efficiently, *sampling-based* algorithms with theoretical guarantees were developed. For example, CELF [19] evaluates the expected spread of nodes with the seed nodes, and select the nodes with the largest marginal spread; TIM [27] counts the frequencies of the nodes appearing in the reversed reachable sets, and chooses the nodes with the highest frequencies; TIM+ [27] is an extension of TIM for large influence graphs.

We say that the above IM algorithms are *offline*, since they are executed on the influence graph once, assuming knowledge of p_{ij} for every i and j. If these values are not known, these algorithms cannot be executed. This problem can be addressed by *online* IM algorithms, as we will discuss next.

4. MAXIMIZING INFLUENCE ONLINE

The goal of the *online influence maximization* (or OIM) is to perform IM without knowing influence probabilities in advance. Given a number N of advertising campaigns (or *trials*), and an advertising budget of k units per trial, we would like to select up to k seed nodes in each trial. These chosen nodes are then advertised or *activated*, and their feedback is used to decide the seed nodes in the next trial. Let us formulate the OIM problem below.

PROBLEM 2. *Given a weighted graph $G=(V,E,p)$ with unknown probabilities p_{uv}, and a budget consisting of N trials with $1\leqslant k\leqslant|V|$ activated nodes per trial, the* online influence maximization *(OIM) problem is to find for each $1\leqslant n\leqslant N$ a set S_n of nodes, with $|S_n|\leqslant k$, such that $\mathbb{E}\left[\left|\bigcup_{1\leqslant n\leqslant N}I(S_n)\right|\right]$ is maximal.*

Note that the IM problem, discussed in Section 3, is a special case of the OIM problem (by setting $N=1$). Since solving the IM

problem is computationally difficult, finding a solution for the OIM is also challenging. We propose a solution that consists of multiple *trials*. In each trial, a *selection* (for choosing appropriate seed nodes) and an *action* (for activating the seed nodes chosen) is performed (Figure 1). The seed selection makes use of one of the offline IM algorithms discussed in Section 3.[1]

We next present the *uncertain influence graph*, which captures the uncertainty of influence probabilities (Section 4.1). We then discuss our solution based on this graph in Section 4.2.

4.1 The Uncertain Influence Graph

We assume that a social network, which describes the relationships among social network users, is given. However, the exact influence probability on each edge is not known. We model this by using the *uncertain influence graph*, in which the influence probabilities of each edges are captured by probability density functions, or *pdf* (Figure 1). The pdf can be refined based on the feedback returned from a trial. Since influence activations are binary random variable, we capture the uncertainty over the influence as a *Beta distribution*. Specifically, the random variable of the influence probability from node i to node j, P_{ij} is modeled as a Beta distribution having probability density function:

$$f_{P_{ij}}(x) = \frac{x^{\alpha_{ij}-1}(1-x)^{\beta_{ij}-1}}{B(\alpha_{ij},\beta_{ij})},$$

where $\mathrm{B}(\alpha_{ij},\beta_{ij})$ is the Beta function, acting as a normalization constant to ensure that the total probability mass is 1, and α_{ij} and β_{ij} are the distribution parameters. For the Beta distribution, $\mathbb{E}[P_{ij}] = \frac{\alpha_{ij}}{\alpha_{ij}+\beta_{ij}}$ and $\sigma^2[P_{ij}] = \frac{\alpha_{ij}\beta_{ij}}{(\alpha_{ij}+\beta_{ij})^2(\alpha_{ij}+\beta_{ij}+1)}$. An advantage of using the Beta distribution is that it is a conjugate prior for Bernoulli distributions, or more generally, binomial distributions. This allows us to compute the posterior distributions easily when new evidence is provided. Section 6 explains this in more detail.

At the time of the first trial, we assume no prior information about the influence graph, except global α and β parameters, shared by all edges, i.e., $P_{ij} \sim \mathrm{B}(\alpha,\beta)\ \forall (i,j) \in E$. These global α and β parameters represent our global *prior belief* of the uncertain influence graph. In the absence of any better prior, we can set $\alpha = \beta = 1$, with $B(1,1)$ being the uniform distribution.

Our model can be extended to handle various prior information about the influence graph. For example, if we have individual prior knowledge (α_{ij},β_{ij}) about an edge, we can set P_{ij} as $P_{ij} \sim \mathrm{B}(\alpha_{ij},\beta_{ij})$. When we have access to only the mean and variance of the influence of an edge, we can derive α_{ij} and β_{ij} from the formulas of $\mathbb{E}[P_{ij}]$ and $\sigma^2[P_{ij}]$ given above. For the situation in which some action logs involving the social network users are available, algorithms for learning the influence probabilities from these logs [11, 12] can be first applied, and the estimated influence probabilities can then be used as prior knowledge for the graph.

4.2 The OIM Framework

Algorithm 1 depicts the solution framework of the OIM problem. In this algorithm, N trials are executed. Each trial involves selecting seed nodes, activating them, and consolidating feedback from them. In each trial n (where $n = 1,\ldots,N$), the following operations are performed on the uncertain influence graph G:

1. `Choose` (Line 5): A seed set S_n is chosen from G, by using an offline IM algorithm, and strategies for handling the uncertainty of G (Section 5).

[1] In this paper we assume that the advertising budget k is fixed for each trial.

Algorithm 1 `Framework(G, k, N)`

1: **Input:** # trials N, budget k, uncertain influence graph G
2: **Output:** seed nodes $S_n(n = 1\ldots N)$, activation results A
3: $A \leftarrow \emptyset$
4: **for** $n = 1$ **to** N **do**
5: $\quad S_n \leftarrow$ `Choose`(G,k)
6: $\quad (A_n,F_n) \leftarrow$ `RealWorld`(S_n)
7: $\quad A \leftarrow A \cup A_n$
8: $\quad$ `Update`(G,F_n)
9: **return** $\{S_n|n = 1\ldots N\}, A$

2. `RealWorld` (Lines 6–7): The selected seeds set is tested in the real world (e.g., sending advertisement messages to selected users in the social network). The feedback information from these users is then obtained. This is a tuple (A_n,F_n) comprised of: (i) the set of activated nodes A_n, and (ii) the set of edge activation attempts F_n, which is a list of edges having either a successful or an unsuccessful activation.
3. `Update` (Line 8): We refresh G based on (A_n,F_n) (Section 6).

One could also choose not to update G, and instead only run an offline IM based on the prior knowledge. Our experimental results show that the influence spread under our OIM framework with proper updates is better than the one without any update. Next, we investigate the design and implementation of `Choose` (Section 5) and `Update` (Section 6).

5. CHOOSING SEEDS

We now study two approaches for selecting k seed nodes in the `Choose` function of Algorithm 1: heuristic-based (Section 5.1) and explore-exploit strategies (Section 5.2).

5.1 Heuristic-Based Strategies

We first discuss two simple ways for choosing seeds from the uncertain influence graph G.

1. `Random`**.** This heuristic, which arbitrarily selects k seed nodes, is based on the *fairness* principle, where every user has the same chance to be activated.

2. `MaxDegree`**.** Given a node p in G, we define the *out-degree* of p to be the number of outgoing edges of p with non-zero influence probabilities. The *out-degree* of p can mean the number of friends of the social network user represented by p, or their number of followers. Intuitively, if p has a higher out-degree, it has a higher chance of influencing other users. The `MaxDegree` heuristic simply chooses the nodes with k highest out-degree values.

The main advantage of these two heuristics is that they are easy to implement. However, they do not make use of influence probability information effectively. In a social network, some users might be more influential than others. It may thus be better to target users with higher influence probabilities on their outgoing edges. The above heuristics also do not consider the feedback information received from the activated users, which can be useful to obtain the true values of the influence probabilities. We will examine a better seed-selection method next.

5.2 Explore-Exploit Strategies

The *Explore-Exploit* (EE) strategy chooses seed nodes based on influence probabilities. Its main idea is to *exploit*, or execute an offline IM algorithm, based on the influence information currently available. Since this information may be uncertain, the seed nodes suggested by *exploit* may not be the best ones. We alleviate this problem by using *explore* operations, in order to improve the knowledge about influence probabilities. Solutions for effectively controlling

explore and exploit operations have been studied in the *multi-armed bandit* (MAB) literature [22, 24]. These MAB solutions inspire our development of the two seed-selection strategies, namely ε*-greedy* and *Confidence-Bound (CB)*. Next, we present these two solutions in detail.

1. ε-greedy. In this strategy (Algorithm 2), a parameter ε is used to control when to explore and when to exploit. Specifically, with probability $1-\varepsilon$, exploitation is carried out; otherwise, exploration is performed.

Algorithm 2 ε-`greedy`(G,k)

1: **Input:** uncertain influence graph $G=(V,E,P)$, budget k
2: **Output:** seed nodes S with $|S|=k$
3: sample z from *Bernoulli*(ε)
4: **if** $z=0$ **then** $S \leftarrow$ `Explore`(G,k)
5: **else** $S \leftarrow$ `Exploit`(G,k)
6: **return** S

In `Exploit`, we execute an offline IM algorithm, given the graph information we have obtained so far. Recall that we model the influence probability p_{ij} between nodes i and j as a probability distribution P_{ij}. We use the mean of P_{ij} to represent p_{ij}, i.e., $p_{ij} = \mathbb{E}[P_{ij}] = \frac{\alpha_{ij}}{\alpha_{ij}+\beta_{ij}}$. A graph with the same node structure but with the p_{ij} values on edges constitutes an influence graph G', on which the offline IM algorithm is executed. Notice that when $\varepsilon = 0$, the solution reduces to exploit-only, i.e., the IM algorithm is run on G' only.

The main problem of `Exploit` is that estimating p_{ij} by $\mathbb{E}[P_{ij}]$ can be erroneous. For example, when P_{ij} is a highly uncertain Beta distribution (e.g., the uniform distribution, $\mathrm{B}(1,1)$), any value in $[0,1]$ can be the real influence probability. Let us consider a node i that has, in reality, a high influence probability p_{ij} on another node j. Due to the large variance in P_{ij}, its value is underestimated. This reduces the chance that `Exploit` chooses node i to activate; consequently, the seed nodes selected may not be the best. The `Explore` routine is designed to alleviate this problem. Rather than equating p_{ij} to $\mathbb{E}[P_{ij}]$, p_{ij} is over-estimated by using P_{ij}'s standard deviation, or σ_{ij}, $p_{ij} = \mathbb{E}[P_{ij}] + \sigma_{ij}$.

Then an offline IM algorithm on these new values of p_{ij} is performed. A node i that has a small chance to be chosen may now have a higher probability to be selected. Our experiments show that the use of `Explore` is especially useful during the first few trials of the OIM solution, since the influence probability values during that time may not be very accurate. From the feedback of activated users, we can learn more about the influence probabilities of the edges of i. We will discuss this in detail in Section 6.

This ε-greedy algorithm has two problems. First, it is difficult to set an appropriate ε, which may have a large impact on its effectiveness. Second, increasing p_{ij} by σ_{ij} may not always be good. Based on these observations, we next propose an improved version of ε-greedy.

2. Confidence-Bound (`CB`). The main idea of this strategy is to use a real-valued parameter θ to control the value of p_{ij}:

$$p_{ij} = \mathbb{E}[P_{ij}] + \theta\sigma_{ij}. \tag{5.1}$$

As shown in Algorithm 3, for every edge e from node i to j, we compute its mean μ_{ij}, variance σ_{ij}, and influence probability p_{ij} based on θ (Lines 3-6). An offline IM algorithm is then run on G', the influence graph with the probabilities computed by Equation 5.1 (Lines 7-8). The set S of seed nodes is then returned (Line 9).

Algorithm 3 `CB`(G,k)

1: **Input:** uncertain influence graph $G=(V,E,P)$, budget k
2: **Output:** seed nodes S with $|S|=k$
3: **for** $e \in E$ **do**
4: $\quad \mu_{ij} \leftarrow \frac{\alpha_{ij}}{\alpha_{ij}+\beta_{ij}}$
5: $\quad \sigma_{ij} \leftarrow \frac{1}{(\alpha_{ij}+\beta_{ij})} \cdot \sqrt{\frac{\alpha_{ij}\beta_{ij}}{(\alpha_{ij}+\beta_{ij}+1)}}$
6: $\quad p_{ij} \leftarrow \mu_{ij} + \theta\sigma_{ij}$
7: $G' \leftarrow G$, with edge probabilities $p_{ij}, \forall (i,j) \in E$
8: $S \leftarrow$ `IM`(G',k)
9: **return** S

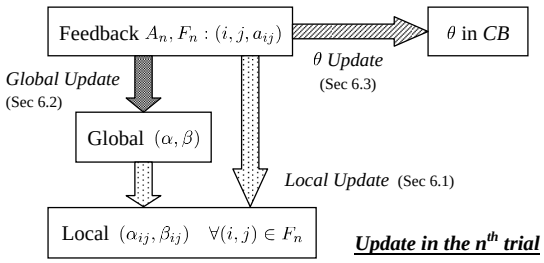


Figure 2: Updating the influence graph and θ with user feedback.

Setting θ. The key issue of Algorithm 3 is how to determine the value of θ, so that the best S can be found. Observe that when $\theta = 0$, p_{ij} becomes μ_{ij} or $\mathbb{E}[P_{ij}]$, and `CB` reduces to `Exploit` of the ε-greedy algorithm. On the other hand, when $\theta = 1$, p_{ij} becomes $\mathbb{E}[P_{ij}] + \sigma_{ij}$, and `CB` is essentially `Explore`. Thus, ε-greedy is a special case of `CB`. However, `CB` does not restrict the value of θ to zero or one. Thus, `CB` is more flexible and general than ε-greedy.

In general, when $\theta > 0$ is used, it means that `CB` considers the influence probabilities given by μ_{ij}'s to be under-estimated, and it attempts to improve the activation effect by using larger values of p_{ij}. On the contrary, if $\theta < 0$, the influence probabilities are considered to be over-estimated, and `CB` reduces their values accordingly. As we will discuss in Section 6.3, θ can be automatically adjusted based on the feedback returned by activated users. This is better than ε-greedy, where the value of ε is hard to set. Note that we choose to use a global θ instead of a local one on each edge, to reduce the number of parameters to be optimized and to improve efficiency.

6. MANAGING USER FEEDBACK

Recall from Algorithm 1 that after the seed nodes S are obtained from `Choose` (Line 5), they are activated in the real world. We then collect *feedback* from the users represented by these nodes (Lines 6–7). The feedback describes which users are influenced, and whether each activation is successful. For instances of such feedback traces, take for example Twitter and other micro-bloggin platforms. In these, the system can track actions such as likes and retweets which are reasonable indicators of influence propagation. We now explain how to use the feedback information to perform `Update` (Line 8), which refreshes the values of influence probabilities and θ used in the `CB` algorithm.

Given a trial n in Algorithm 1, let A_n be the set of activated nodes in that trial, and F_n be the set of activation results. Specifically, F_n contains tuples in the form of (i,j,a_{ij}), where i and j are users between which an activation was attempted; $a_{ij} = 1$ if the influence was successful, and $a_{ij} = 0$ otherwise. Note that (i,j) is an edge of

the influence graph G. Also, F_n might not contain all edges of G, since an activation might not reach every user in G.

Three kinds of updates can be performed based on A_n and F_n:

1. Local (Section 6.1): Update the influence probability's distribution (i.e., $\mathrm{B}(\alpha_{ij},\beta_{ij})$) if the edge (i.e., activation from i to j) was attempted;

2. Global (Section 6.2): Update the global prior information α and β, which are shared by all edges of G; and

3. θ (Section 6.3): Update the value of θ used in `CB`, if it is used as a seed selection strategy in `Choose`.

Figure 2 illustrates these three kinds of updates in the n-th trial. In the next sections, we discuss how to conduct these updates in detail. We remark that these update methods do not affect `Random` and `MaxDegree`, since they do not use these updated values.

6.1 Updating Local $\vec{\alpha}$ and $\vec{\beta}$

As we mentioned before, the influence probability between any two adjacent nodes i and j is modeled as a Beta distribution with parameters α_{ij} and β_{ij}, denoted as $P_{ij} \sim \mathrm{B}(\alpha_{ij},\beta_{ij})$. Since the Beta distribution is a conjugate prior for the Bernoulli distribution, then, given feedback (i,j,a_{ij}) in F_n (seen as a Bernoulli trial), we can update the distribution as follows:

1. If $a_{ij}=1$, i.e., the activation from node i to node j was successful: $P_{ij} \sim \mathrm{B}(\alpha_{ij}+1,\beta_{ij})$;

2. If $a_{ij}=0$, i.e., the activation from node i to node j failed: $P_{ij} \sim \mathrm{B}(\alpha_{ij},\beta_{ij}+1)$.

In the beginning, we have no prior information about the distribution except the global α and β, i.e., $\alpha_{ij}=\alpha$ and $\beta_{ij}=\beta$. After n trials and activations, we have thus collected n pieces of feedback information. Let h_{ij} (m_{ij}) be the number of successful (failed) activations for edge (i,j). Then $\alpha_{ij}=\alpha+h_{ij}, \quad \beta_{ij}=\beta+m_{ij}$. Hence, this local update is equivalent to maintaining a distribution $\mathrm{B}(\alpha+h_{ij},\beta+m_{ij})$, i.e., the distributions on the edges simply count the number of successful and failed activations passing through that edge, smoothed by the prior $\mathrm{B}(\alpha,\beta)$.

Note that this update process corresponds exactly to the MLE approach taken by [11] to learn influence probabilities from action logs, with a smoothing prior added. The important difference is that [11] only conducts this estimation for edges where there is evidence, i.e., local updates. If the evidence is sparse, this can lead to a sub-optimal, and overfitting, influence graph. Global update of Beta priors, which go beyond the local feedback, can yield a better influence graph.

6.2 Updating Global $\vec{\alpha}$ and $\vec{\beta}$

Local updates to the random variable P_{ij} allows the edge influence probability distribution to be updated directly. In the first few trials, however, the real influence spread is sparse and limited, and most of the edges will not be reached by an activation. Therefore, the influence of choosing a good prior will weigh heavily on how `Choose` performs. Once some evidence is gathered, this prior can be refined by taking into account the feedback in a *global* sense, over all trials up to the current one. Next, we present two methods of updating the global α and β priors based on the feedback.

Least Squares Estimation. The first solution is to find the best fit for the α and β priors according to the real spread that we obtained from the real world test at each trial.

Let us first explain the reasoning when there is one seed node (i.e., $|S_n|=1$), and we fix $\alpha=1$. Let $\mathcal{A}_n$ be the set of successful activated nodes before the n-th trial (i.e., $\mathcal{A}_n=\cup_{l=1}^{n-1}A_l$), and $\sigma_n(\{i\})$ be the expected number of additional activated nodes (or expected additional spread) from the seed node i in the n-th trial. For $S_n=\{s\}$, $\sigma_n(\{s\})$ is:

$$\sigma_n(\{s\}) = 1 + \sum_{\substack{(s,i)\in E\\ i\notin\mathcal{A}_n}} p_{si}\times\sigma_n(\{i\}) + \sum_{\substack{(s,i)\in E\\ i\in\mathcal{A}_n}} p_{si}\times(\sigma_n(\{i\})-1),$$

which is the sum of the outgoing spreads weighted by the outgoing probabilities p_{si} and discounted by 1 for nodes already activated along an outgoing edge.

We estimate $\sigma_n(\{s\})$ by $|A_n|$ from the feedback obtained by the influence campaign. We also estimate $p_{si}=\frac{\alpha+h_{si}}{\alpha+h_{si}+\beta+m_{si}}$, i.e., the mean of $\mathrm{B}(\alpha_{ij},\beta_{ij})$. Note that $h_{si}+m_{si}$ is the total number of attempts from node s to i, which is the same for neighbors of s because every activation through s tries to activate all outgoing nodes in the independent cascade model. Thus, we use t_s to denote $h_{si}+m_{si}\ \forall(s,i)\in E$. By further estimating $\sigma_n(\{i\})$ by an overall estimation $\hat{\sigma}_n$ and set $\alpha=1$, we obtain

$$|A_n| = 1 + \frac{1}{\beta+t_s+1}\left(\sum_{(s,i)\in E}(h_{si}+1)\hat{\sigma}_n - \sum_{(s,i)\in E, i\in\mathcal{A}_n}(h_{si}+1)\right).$$

Let o_s be the outgoing degree of s, a_s be the number of (previously) activated neighbors of s (i.e., $a_s=|\{i|(s,i)\in E\wedge i\in\mathcal{A}_n\}|$), h_s be the number of total successful activations (or hits) on outgoing edges of s, and h_{as} be the number of total hits on edges leading to activated neighbors. The above equation is simplified to

$$(|A_n|-1)\beta = (1-|A_n|)(t_s+1) + (h_s+o_s)\hat{\sigma}_n - (h_{as}+a_s).$$

We then rewrite it as the form of $x_n\beta=y_n$. Since this equation also applies to activations in all trials up to the current one, we use the least square estimator for linear regression without an intercept term to get an estimator for β, $\hat{\beta}=(\vec{x}\cdot\vec{y})\ /\ (\vec{x}\cdot\vec{x})$, where $\vec{x}$ and $\vec{y}$ are the vectors of values x_n and y_n. The same principles apply when estimating α and β simultaneously, and we omit the details here.

We estimate $\hat{\sigma}_n$ by the average spread of the node from the activation campaigns, i.e., $\hat{\sigma}_n=\sum_{l=1}^{n}|A_n|/\sum_{l=1}^{n}|S_n|$. Note that, when $\hat{\sigma}_n=0$, the equation for $|A_n|$ is exactly the degree discount estimator from the IM literature [7], and represents a lower bound on the spread from a node.

A further complication occurs when $|S_n|>1$, which might result in an equation at least quadratic in β, due to the influence probability equations of nodes which are neighbors of more than one seed node. In this work, we simplify the estimation by assuming full independence among seed nodes, and hence replacing x_n and y_n by the sum over all $s\in S_n$.

We remark that the estimator above suffers from the reliance on the spread estimation $\hat{\sigma}_n$. However, it is a good option when we cannot access the full activation feedback F_n, but instead, do have the access to the set of successful activated nodes in each trial (i.e., the set A_n). This may happen in an alternate problem setting when one cannot get all the feedback information from the activated users in A_n.

Maximum Likelihood Estimation. Given the feedback from trial n, we can compute the likelihood of the feedback F_n given the probabilities of each edge in the feedback tuples, by assuming that they are independently activated. The likelihood depends on the successful activations (hits) and failed activations (misses) of each edges and the global prior parameters α and β:

$$\mathcal{L}(F_n) = \prod_{(i,j,a_{ij})\in F_n} p_{ij}^{a_{ij}}(1-p_{ij})^{1-a_{ij}},$$

$$\mathcal{L}(F_n \mid \alpha,\beta) = \prod_{(i,j,a_{ij})\in F_n} \frac{(\alpha+h_{ij})^{a_{ij}}(\beta+m_{ij})^{1-a_{ij}}}{\alpha+\beta+h_{ij}+m_{ij}}.$$

We need to find the parameters α and β for maximizing the likelihood:

$$\arg\max_{\alpha,\beta} \mathcal{L}(F_n \mid \alpha, \beta).$$

To simplify calculations, we use the usual method of taking the maximum of the log likelihood, and arrive at the optimal values by solving the equations $\frac{\partial \log \mathcal{L}(F_n|\alpha,\beta)}{\partial \alpha} = 0$ and $\frac{\partial \log \mathcal{L}(F_n|\alpha,\beta)}{\partial \beta} = 0$ for α and β, respectively. This becomes:

$$\sum_{(i,j,a_{ij})\in F_n, a_{ij}=1} \frac{1}{\alpha + h_{ij}} = \sum_{(i,j,a_{ij})\in F_n, a_{ij}=0} \frac{1}{\beta + m_{ij}}.$$

This equation can be solved numerically by setting α and solving β. In practice, we can fix $\alpha = 1$, and solve β by binary search. The full details can be found in our technical report [18].

6.3 Updating $\vec{\theta}$

We now explain how to dynamically update the value of θ used in the CB strategy (Section 5.2).

Let $\vec{\theta} = \{\theta_1, \theta_2, \ldots, \theta_q\}$ be the q possible values of θ. We also let $\vec{\varphi} = \{\varphi_1, \varphi_2, \ldots, \varphi_q\}$, where φ_j is the probability of using θ_j in CB. Initially, $\varphi_j = 1/q$ for $j = 1, \ldots, q$, and its value is updated based on the gain obtained in each trial. The gain is defined as $G_n = |A_n|/|V|$, where $|A_n|$ is the real influence spread observed in each round. We then determine $\vec{\theta}$ by using the exponentiated gradient algorithm [5]. The rationale of using this solution is that if the value of θ_j used in this trial results in a high gain, the corresponding φ_j will be increased by the algorithm, making θ_j more likely to be chosen in the next trial. Algorithm 4 gives the details.

Algorithm 4 `ExponentiatedGradient`$(\vec{\varphi}, \delta, G_n, j, \mathbf{w})$

1: **Input:** $\vec{\varphi}$, probability distribution; δ, accuracy parameter; G_n, the gain obtained; j, the index of latest used θ_j; $\mathbf{w}$, a vector of weights; N, the number of trials.
2: **Output:** θ
3: $\gamma \leftarrow \sqrt{\frac{\ln(q/\delta)}{qN}}$, $\tau \leftarrow \frac{4q\gamma}{3+\gamma}$, $\lambda \leftarrow \frac{\tau}{2q}$
4: **for** $i = 1$ **to** q **do**
5: $\quad w_i \leftarrow w_i \times \exp\left(\lambda \times \frac{G_n \times \mathbb{I}[i=j]+\gamma}{\varphi_i}\right)$
6: **for** $i = 1$ **to** q **do**
7: $\quad \varphi_i \leftarrow (1-\tau) \times \frac{w_i}{\sum_{j=1}^{k} w_j} + \tau \times \frac{1}{q}$
8: **return** sample from $\vec{\theta}$ according to $\vec{\varphi}$ distribution

Here, γ and λ are smoothing factors used to update weights, and $\mathbb{I}[z]$ is the indicator function. We compute $\vec{\varphi}$ by normalizing vector $\mathbf{w}$ with regularization factor τ. All the values in $\mathbf{w}$ are initialized with the value of 1.

In [5], it is shown that, for a choice of constant θ's, `ExponentiatedGradient` can provide a regret bound on the optimal sequence of chosen θ in the vector. In our case, the experimental results also show that `ExponentiatedGradient` is the best performing strategy.

7. INCREMENTAL SOLUTION FOR OIM

In our OIM solution, an offline IM algorithm is invoked once in every trial to select seeds. However, the state-of-the-art IM algorithms with good theoretical approximation bounds, such as CELF, TIM, and TIM+, are generally costly to run, especially for large graphs with high influence probabilities. For instance, in our experiments in DBLP, which has around two million edges, the best IM algorithm, TIM+, takes 30 minutes to select the nodes for a trial. Since an OIM solution requires multiple trials, the running time can be very high in practice. In this section, we study how to improve the scalability of the OIM solution. We use TIM+ as an example, but our approach can generally be used to handle other IM algorithms (e.g., CELF) that perform sampling on influence graphs.

Let us first explain the intuition behind our approach. We observe that the running time of TIM+ is dominated by the cost of sampling the influence graph. In our experiments, over 99% of its computation is spent in generating samples (which are *random reverse reachable sets*, or RR sets in short). Given a node v, the RR is the set of all nodes that can influence v. Another observation is that the size of the real-world feedback F_n is relatively small compared with the number of edges in influence graph G. In DBLP, when k is set to 1 for each trial, the average size of F_n is less than 1% of the total number of edges of G. Since samples are generated from G, and the user feedback only influences a small part of G, it can only affect a few samples obtained from the updated G in the next trial. Based on these observations, we develop a new method to save the sampling effort, based on reusing samples of previous trials. We call this the *incremental* algorithm of OIM.

Here we give an outline of this approach; the details can be found in [18]. The algorithm stores the samples generated during the TIM+ computation in previous trials in a data structure. In a new trial, when TIM+ requires a new sample, instead of immediately sampling G, we first randomly select a stored sample from the data structure, and check whether this sample can be reused. If this is the case, the selected sample will be returned; otherwise, a new sample will be generated according to the current G and stored in the data structure. Next, we discuss how to determine whether a sample can be reused.

Intuitively, if G remains the same (or only deviates little), the occurrence probability of a sample is only slightly affected. Hence, reusing a randomly selected sample will not incur much error. Recall that in our OIM framework, only the update component might affect the estimation of the influence graph. Therefore, we only need to check whether the updates affect G or the samples significantly, in order to determine whether a sample can be reused. Suppose s is the sample to be checked. In TIM+, s is a random RR set. We design three checks corresponding to three updates in Section 6.

[1. Local check] If all edges that point to a node in s are not affected by local updates, local check is passed.

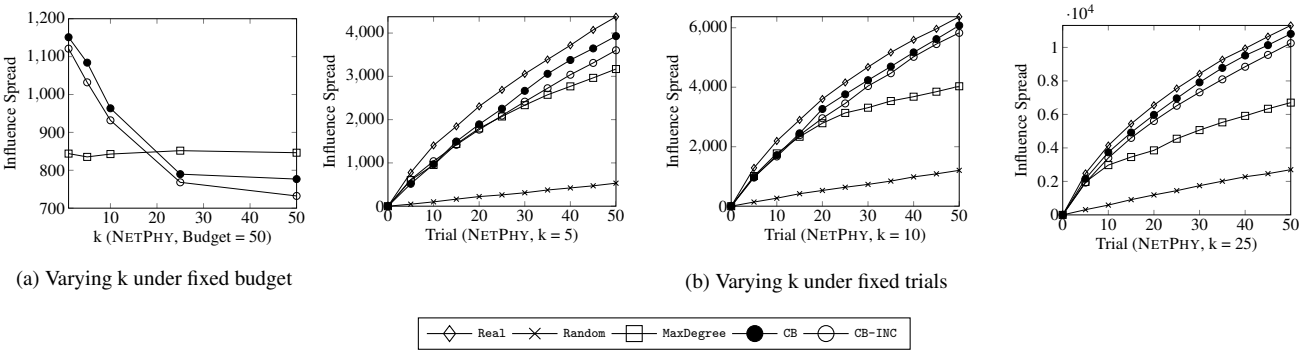
[2. Global check] If the prior of generating s (denoted by α^t, β^t) is close to the current prior, global check is passed. The check is controlled by a threshold τ, i.e., the condition is $|\frac{\alpha^t}{\alpha^t+\beta^t} - \frac{\alpha}{\alpha+\beta}| < \tau$.

[3. θ check] If θ and the standard deviation of the global prior when generating s (denoted by θ^t, σ^t) is similar to the current values, θ check is passed, i.e., $|\theta^t\sigma^t - \theta\sigma| < \tau$.

We remark that with our checking mechanism, conducting checks on a sample is about d times faster than sampling a new one, where d is the average in-degree for a node. As shown by our experiments, this incremental approach can significantly save the computation effort of our OIM solution.

8. EXPERIMENTAL EVALUATION

Setup. We developed a "real-world simulator" to mimic the user feedback process of Figure 1. This simulator first uses a real social network to obtain a graph G. It then associates an influence probability to each edge in G, where $p_{ij} = 1/d_j$, with d_j the in-degree of node j. This setting of influence probability values is adopted in previous works [6, 7, 12, 16, 19, 27].



(a) Varying k under fixed budget

(b) Varying k under fixed trials

Figure 3: Heuristic-based v.s. Explore–Exploit.

Table 2: Datasets

Dataset	NETHEPT	NETPHY	DBLP
# of Nodes	15K	37K	655K
# of Edges	59K	231K	2.1M
avg. degree	7.73	12.46	6.1
max. degree	341	286	588

When the chosen seed nodes are tested on whether they can influence other nodes, the simulator runs a single independent cascade simulation on G, and obtains feedback information F_n, in a form of (i, j, a_{ij}) and A_n, the set of successfully activated nodes. We measure the effectiveness of an OIM solution by its influence spread in the real world, after N trials, as the total number of successfully activated nodes in these trials, i.e, $|\cup_{n=1}^{N} A_n|$. We repeat each solution 10 times and report the average.

Datasets. We have studied several real social network datasets. We have used NETHEPT and NETPHY are collaboration networks, obtained from arXiv.org in the High Energy Physics Theory and Physics domains, respectively. We have also used the DBLP graph, which is an academic collaboration network. In these datasets, nodes represent authors, and edges representing co-authorship. These datasets are commonly used in the literature of influence maximization [6, 7, 12, 16, 27]. Table 2 shows the details of these datasets.

Options for OIM algorithm. We have evaluated several possible options for the *seed selection* and *graph update* components for our OIM solution:

[Choosing seeds]

- Heuristic-based strategies: `Random`, `MaxDegree`;
- Explore–Exploit strategies: 1) `Exploit` contains only exploit algorithm; 2) ε`-greedy` represents ε-greedy algorithm; 3) `CB` is our Confidence-Bound explore–exploit algorithm with Exponentiated Gradient update.

[Updating graph]

- `NO` does not conduct any update;
- `LOC` only local updates;
- `LSE` local and global updates where Least Squares Estimation is adopted in global update;
- `MLE` as `LSE`, but Maximum Likelihood Estimation is adopted.

In our experiments, we compare the algorithms using combinations of the above two components. Note that `Random` and `MaxDegree` do not rely on the influence probability of the edges, and they are not combined with update methods. When a particular EE strategy is adopted, the update method would be specified, for instance, `CB+MLE` means that we use `CB` with `MLE` update. By default, we use `MLE` for updating the graph. Furthermore, if the EE strategy is used in choosing seeds, we use `CB` by default.

When an IM algorithm is invoked in an EE strategy, we use TIM+ since it is the state-of-art influence maximization algorithm. We also compare the incremental approach with the non-incremental one for EE strategy. For example, we denote the incremental version for `CB` as `CB-INC`.

Parameters. By default, the global prior is set to be $\mathrm{B}(1, 19)$, $\theta = \{-1, 0, 1\}$ in `CB`, $\varepsilon = 0.1$ in ε`-greedy`, and $\tau = 0.02$ in the incremental approach.

Our algorithms, implemented in C++, are conducted on a Linux machine with a 3.40 GHz Octo-Core Intel(R) processor and 16GB of memory. Next, we focus on NETPHY, and evaluate different combinations of the algorithms in our OIM framework. We summarize our results for other datasets in Section 8.2.

8.1 Results on NetPHY

Heuristic-based v.s. Explore–Exploit. We first fix the total budget and verify how the OIM algorithms perform with different number of trials. We set $Budget = 50$, and vary k in $\{1, 5, 10, 25, 50\}$. By varying k, we essentially vary the total budget. For example, with $k = 5$, 50 units of budget is invested over $N = 10$ trials. Figure 3a shows our results. Since `Random` only has influence spread less than 200 on average, we do not plot it. We observe that the spread of `MaxDegree` does not change much since it does not depend on the real-world feedback. For `CB`, its spread increases when k decreases and it is better than `MaxDegree` when $k \leqslant 10$ (or $N \geq 5$). Specifically, when $k = 1$, `CB` is about 35% better than `MaxDegree`. The reason is that, for `CB`, a smaller k indicates more chances to get real-world feedback, and thus, more chances to learn the real influence graph, which leads to a better result. Moreover, when $k = 50$, all budget is invested once, which can be regarded as an offline solution, and produces the worst result for `CB`. This further indicates the effectiveness of our OIM framework. For `CB-INC`, it performs close to `CB` with only a small discount (around 5% for different k) on the spread. It supports our claim that the incremental approach can perform without incurring much error.

We next fix k and compare different algorithms in Figure 3b. The results are consistent with our previous findings that `CB` outperforms other variants. `CB-INC` produces similar results with `CB`. We observe that the gap between `CB` and `MaxDegree` increases with N and k. For example, at $N = 50$, `CB` is about 20% better than `MaxDegree` when $k = 5$, and the percentage grows to 45% when $k = 25$. The reason is that larger k and larger N give more chances for `CB` to learn the real influence graph. We also plot the result for TIM+ when the real influence probability is known, denoted as `Real`. This can be seen as an oracle, serving as a reference for other algorithms. We

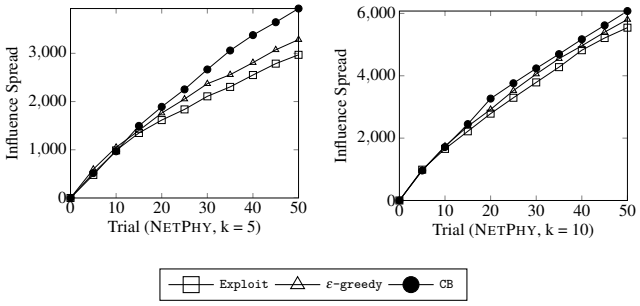
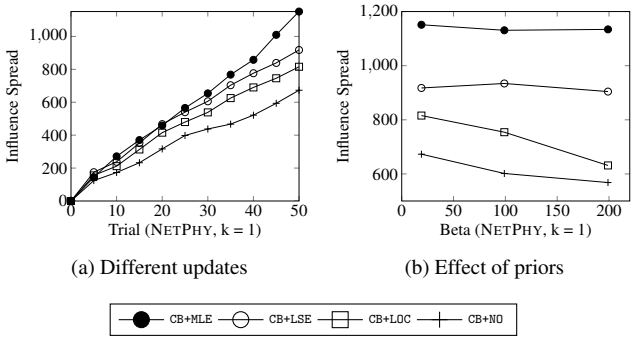


Figure 4: Explore–exploit strategies



(a) Different updates (b) Effect of priors

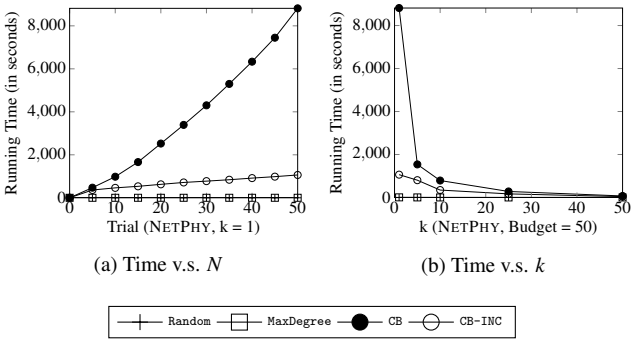
Figure 5: Comparing different updating methods

find that CB performs close to Real, and its discount on the spread decreases with N. For example, when $k = 5$, the discount decreases from 30% at $N = 10$ to 13% at $N = 50$. This indicates that, with more real-world feedback, the learned graph for CB is closer to the real graph, and thus, leads to a closer result to Real.

Explore–Exploit Strategies. We compare three versions of the EE strategies for different k in Figure 4. We observe that Exploit is the worst, since it may suffer from the wrong prediction of the influence probabilities and does not explore other potential high influencing nodes. CB is the best, especially, for small k. When $k = 5, N = 50$, CB is about 20% and 32% better than ε-greedy and Exploit, respectively. The reason is that for a smaller k, fewer feedback tuples are returned in one trial, which makes the learned influence graph converge to the real graph slower. Hence, the effect of exploration is strengthened, which is more favorable to CB. We have also conducted experiments for ε-greedy by varying ε. We observe that its performance is sensitive to ε and $\varepsilon = 0.1$ is the best one in our results, but it is still worse than CB in all cases.

Updating the uncertain influence graph. In Figure 5a, we compare different updating methods for the uncertain influence graph. Although NO makes use of the prior knowledge about the influence graph to select seeds, it still performs worse than other update options. LOC is slightly better, but still worse than MLE and LSE, since it does not employ any global update and it suffers from the sparseness of the activations. MLE is the best (about 25% better than LSE and 40% better than LOC), which is consistent with the fact that MLE makes use of the full feedback to update the graph while LSE only utilizes the set of successfully activated nodes.

We also test the updating methods with different priors (Figure 5b) to check whether they are sensitive to the prior. We observe that while LOC and NO fluctuate a lot with different priors, MLE and LSE's performance is very stable. In fact, during different runs of MLE and LSE with different priors, the global β values all converge to around 27. This supports the fact that the global updating techniques are



(a) Time v.s. N (b) Time v.s. k

Figure 6: Cumulative running time

crucial when we do not have good prior information. Even an inexact choice of prior will be generally fixed, minimizing the impact on performance.

Efficiency. In Figure 6a, we illustrate the cumulative running time for running N trials for different algorithms. Random and MaxDegree are most efficient as they do not rely on any influence evaluation. With the help of incremental approach, CB-INC runs significantly faster than CB, and for the case where $N \geqslant 10$, it achieves about 10 times speedup. For instance, at $N = 50$, CB-INC reduces the running time by 88%, compared to CB. This is intuitive, as in the first few trials the graph is more uncertain, and the updates affect the samples a lot. However, when $N \geqslant 10$, we observe that the global priors become more stable, leading to a high ratio of re-using samples (e.g., the ratio is about 80% to 99% when $N \geqslant 10$). Moreover, the average in-degree of NETPHY is 12.46, making the time of generating a new sample about an order of magnitude slower than re-using a sample. These two factors together make CB-INC have a much more efficient performance than CB.

We then show the efficiency results by fixing $Budget = 50$ and varying k in Figure 6b. The running time of MaxDegree and Random is stable for various k, while CB and CB-INC show a decline on efficiency when k decreases. This is because a smaller k indicates that more trials are required to invest all budget, and so, TIM+ should be executed more often, for a general decrease in efficiency. Another observation is that the improvement of CB-INC over CB increases with k. This further strengthens the utility of using CB-INC in practice. Figure 6b and Figure 3a together show a tradeoff of setting k: a smaller k leads to a better performance in spread but worse performance in efficiency. We suggest to set a small k to ensure the algorithm's better performance in spread. The value of k will depend on how much total time that the user can afford.

Effect of τ. We also verify the effect of τ in the incremental approach by varying τ from 0.01 to 0.03 and fixing $k = 1, Budget = 50$. We compare them with CB, the non-incremental algorithm. First, a smaller τ gives better results in terms of influence spread. For instance, it leads to 3%, 5%, 15% discount in spread compared with CB for $\tau = 0.01, 0.02, 0.03$, respectively. However, a smaller τ leads to a slowdown in efficiency since it has a stricter requirement in global check. For example, the running time for $\tau = 0.01$ is about 28% slower than the one for $\tau = 0.02$ and 38% worse than the one for $\tau = 0.03$.

Discussion. The OIM framework is highly effective in maximizing influence when the real influence probabilities are unknown. In this framework, MLE is the best updating method. Moreover, CB and CB-INC consistently outperform other algorithms. By using CB-INC, we can also significantly improve the efficiency of CB, with only a small discount in influence spread.

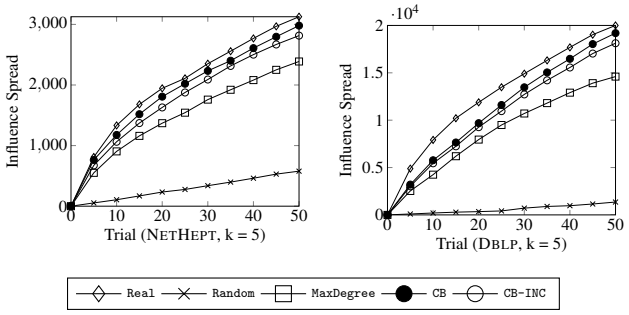


Figure 7: Effectiveness on other datasets

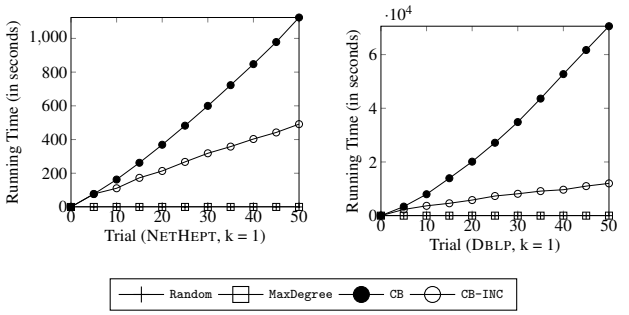


Figure 8: Efficiency on other datasets

8.2 Results for NetHEPT and DBLP

Figure 7 and Figure 8 show representative results for NETHEPT and DBLP. These results are consistent with the ones for NETPHY, where CB and CB-INC are close to the oracle (Real), and better than heuristic-based algorithms in maximizing influence spread. For efficiency, CB-INC significantly reduces the running time of CB, especially for a large dataset DBLP. For instance, at $k = 1, N = 50$, CB-INC saves 16 hours compared with CB which costs 19 hours in total to get the result for DBLP.

9. CONCLUSIONS

In this paper, we examine how to perform influence maximization when influence probabilities may not be known in advance. We develop a new solution, where IM is performed in multiple trials, and we have proposed explore–exploit strategies for this problem. We showed experimentally that explore–exploit based on the uncertainty in the graph performs well. We also proposed novel methods to update the knowledge of the graph based on the feedback received from the real world, and showed experimentally that they are effective in longer campaigns. Even when the influence probabilities are not known in advance, the influence spread of our solution is close to the spread using the real influence graph, especially when the number of trials increases.

In the future, we will examine the scenario where budgets are different in each trial. We will extend our solution to handle other complex situations (e.g., the change of influence probability values over time), consider IM methods (e.g., [28], [2]) that utilize community and topic information, and other influence propagation models, such as linear threshold or credit distribution [12, 14, 26]. Another direction is to increase the scalability of our methods; this may require distributed algorithm, such as distributed sampling.

Acknowledgments. Siyu Lei, Silviu Maniu, Luyi Mo, and Reynold Cheng were supported by University of Hong Kong (201311159095 and 201411159171). We thank the reviewers for their comments.

10. REFERENCES

[1] S. Agrawal and N. Goyal. Analysis of thompson sampling for the multi-armed bandit problem. In *COLT*, 2012.
[2] C. Aslay, N. Barbieri, F. Bonchi, and R. A. Baeza-Yates. Online topic-aware influence maximization queries. In *EDBT*, 2014.
[3] P. Auer. Using confidence bounds for exploitation-exploration trade-offs. *JMLR*, 3:397–422, 2003.
[4] C. Borgs, M. Bratbar, J. Chayes, and B. Lucier. Maximizing social influence in nearly optimal time. In *SODA*, 2014.
[5] N. Cesa-Bianchi and G. Lugosi. *Prediction, Learning, and Games*. Cambridge University Press, 2006.
[6] W. Chen, C. Wang, and Y. Wang. Scalable influence maximization for prevalent viral marketing in large-scale social networks. In *KDD*, 2010.
[7] W. Chen, Y. Wang, and S. Yang. Efficient influence maximization in social networks. In *KDD*, 2009.
[8] W. Chen, Y. Wang, and Y. Yuan. Combinatorial multi-armed bandit: General framework and applications. In *ICML*, 2013.
[9] W. Chen, Y. Yuan, and L. Zhang. Scalable influence maximization in social networks under the linear threshold model. In *ICDM*, 2010.
[10] P. Domingos and M. Richardson. Mining the network value of customers. In *KDD*, 2001.
[11] A. Goyal, F. Bonchi, and L. V. Lakshmanan. Learning influence probabilities in social networks. In *WSDM*, 2010.
[12] A. Goyal, F. Bonchi, and L. V. Lakshmanan. A data-based approach to social influence maximization. *PVLDB*, 5(1), 2011.
[13] A. Goyal, W. Lu, and L. V. Lakshmanan. Celf++: Optimizing the greedy algorithm for influence maximization in social networks. In *WWW*, 2011.
[14] J. Huang, X.-Q. Cheng, H.-W. Shen, T. Zhou, and X. Jin. Exploring social influence via posterior effect of word-of-mouth recommendations. In *WSDM*, 2012.
[15] K. Jung, W. Heo, and W. Chen. Irie: Scalable and robust influence maximization in social networks. In *ICDM*, 2012.
[16] D. Kempe, J. Kleinberg, and E. Tardos. Maximizing the spread of influence through a social network. In *KDD*, 2003.
[17] J. Kim, S.-K. Kim, and H. Yu. Scalable and parallelizable processing of influence maximization for large-scale social networks? In *ICDE*, 2013.
[18] S. Lei, S. Maniu, L. Mo, R. Cheng, and P. Senellart. Online influence maximization (extended version). arXiv:1056.01188, 2015.
[19] J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. VanBriesen, and N. Glance. Cost-effective outbreak detection in networks. In *KDD*, 2007.
[20] M. J. Lovett, R. Peres, and R. Shachar. On brands and word of mouth. *J. Marketing Research*, 50(4), 2013.
[21] W. Lu, F. Bonchi, A. Goyal, and L. V. Lakshmanan. The bang for the buck: Fair competitive viral marketing from the host perspective. KDD, 2013.
[22] S. Richard and A. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.
[23] M. Richardson and P. Domingos. Mining knowledge-sharing sites for viral marketing. In *KDD 2002*.
[24] H. Robbins. Some aspects of the sequential design of experiments. *Bull. Amer. Math. Soc.*, 58(5), 1952.
[25] K. Saito, R. Nakano, and M. Kimura. Prediction of information diffusion probabilities for independent cascade model. In *KES*, 2008.
[26] Y. Singer. How to win friends and influence people, truthfully: influence maximization mechanisms for social networks. In *WSDM*, 2012.
[27] Y. Tang, X. Xiao, and Y. Shi. Influence maximization: Near-optimal time complexity meets practical efficiency. In *SIGMOD*, 2014.
[28] Y. Wang, G. Cong, G. Song, and K. Xie. Community-based greedy algorithm for mining top-k influential nodes in mobile social networks. In *KDD*, 2010.