Online Topic-Aware Influence Maximization

Shuo Chen[†] Ju Fan[‡] Guoliang Li[†] Jianhua Feng[†] Kian-lee Tan[‡] Jinhui Tang[*]
[†]*Department of Computer Science, TNList, Tsinghua University, Beijing, China.*
[‡]*School of Computing, National University of Singapore, Singapore.*
[*]*School of Computer Science, Nanjing University of Science and Technology, Nanjing, China.*
{liguoliang,fengjh}@tsinghua.edu.cn; s-chen13@mails.thu.edu.cn;
{fanj, tankl}@comp.nus.edu.sg; jinhuitang@njust.edu.cn

ABSTRACT

Influence maximization, whose objective is to select k users (called seeds) from a social network such that the number of users influenced by the seeds (called influence spread) is maximized, has attracted significant attention due to its widespread applications, such as viral marketing and rumor control. However, in real-world social networks, users have their own interests (which can be represented as topics) and are more likely to be influenced by their friends (or friends' friends) with similar topics. We can increase the influence spread by taking into consideration topics. To address this problem, we study topic-aware influence maximization, which, given a topic-aware influence maximization (TIM) query, finds k seeds from a social network such that the topic-aware influence spread of the k seeds is maximized. Our goal is to enable online TIM queries. Since the topic-aware influence maximization problem is NP-hard, we focus on devising efficient algorithms to achieve instant performance while keeping a high influence spread. We utilize a maximum influence arborescence (MIA) model to approximate the computation of influence spread. To efficiently find k seeds under the MIA model, we first propose a best-effort algorithm with $1-\frac{1}{e}$ approximation ratio, which estimates an upper bound of the topic-aware influence of each user and utilizes the bound to prune large numbers of users with small influence. We devise effective techniques to estimate tighter upper bounds. We then propose a faster topic-sample-based algorithm with $\epsilon \cdot (1-\frac{1}{e})$ approximation ratio for any $\epsilon \in (0, 1]$, which materializes the influence spread of some topic-distribution samples and utilizes the materialized information to avoid computing the actual influence of users with small influences. Experimental results show that our methods significantly outperform baseline approaches.

1. INTRODUCTION

The influence maximization problem seeks to select k users from a social network through whom the number of influenced users is maximized. The omnipresent, successful social networks, like Facebook and Twitter, have boosted research on the influence maximization problem due to its potential commercial value, such as viral marketing [2], rumor control, and information monitoring [20, 11, 15].

Evidently, in real-world social networks, users have their own interests (which can be represented as topics) and are more likely to be influenced by their friends (or friends of friends) with similar interests. For example, a soccer fan will more likely be influenced by famous soccer players rather than basketball players. We can increase the influence spread by taking into account topics in influence maximization [1]. To address this problem, we study topic-aware influence maximization. We first model a social network as a topic-aware graph, where the edge of any two users is associated with a topic distribution. For example, given an edge from user u to user v, a topic distribution $\langle$`soccer`:0.6, `health`:0.7, `high-tech`:0.8$\rangle$ on the edge means that the probabilities of v influenced by u on topics `soccer`, `health`, and `high-tech` are respectively 0.6, 0.7, and 0.8. Then, given a topic-aware influence maximization (TIM) query, e.g., $\mathcal{Q}$=($\langle$`soccer`:0.4, `high-tech`:0.6$\rangle$, k), we find k seeds such that the topic-aware influence spread of these k seeds with respect to the TIM query is maximized.

There are many real-world applications for topic-aware influence maximization, e.g., topic-aware advertisement and topic-aware rumor control. For example, a startup company on `e-health` wants to pay users to advertise on a social network. Due to budget constraints, it can only afford to pay k users, e.g., $k = 100$, and it wants to maximize the benefit (i.e., the topic-aware influence spread) given the k selected users. Obviously the company only cares about the users with interests in `health` and `high-tech`, and thus it can issue a TIM query $\mathcal{Q} = (\langle$`health`:0.8, `high-tech`:0.2$\rangle, 100)$.

The difference from traditional influence maximization is that the influence between two users depends on not only the social relationships but also the topic distributions on the social links. There are three main challenges arising in the topic-aware influence maximization problem. The first is how to obtain the topic distributions. Thanks to the user-generated content on social networks like Twitter or Weibo, we can run text-based topic discovery algorithms to generate the topic distributions [1]. The second is how to compute the topic-aware influence spread. We employ a tree-based model to compute the topic-aware influence by considering both the social links and the topic distributions (see Section 2.2). The third is to achieve high performance. Since there are a large number of TIM queries on a social network and each query wants to be answered instantly, our goal

Proceedings of the VLDB Endowment, Vol. 8, No. 6

is to enable online TIM queries. However, the topic-aware influence maximization problem is NP-hard and computing the topic-aware influence spread is #P-hard [1]. Thus, in this paper we focus on devising efficient greedy algorithms to achieve real-time performance while keeping a high influence spread with a theoretical guarantee.

To achieve instant performance for answering a TIM query, we utilize a maximum influence arborescence (MIA) model to effectively approximate the computation of influence spread. Under the MIA model, we propose a best-effort algorithm with $1-\frac{1}{e}$ approximation ratio, which estimates an upper bound of the topic-aware influence of each user and utilizes the bound to prune a large number of users with small influence. We develop effective techniques to accurately estimate upper bounds. To further improve the performance, we propose a faster topic-sample-based algorithm with $\epsilon \cdot (1-\frac{1}{e})$ approximation ratio for any $\epsilon \in (0, 1]$, which materializes the influence of some topic-distribution samples and utilizes the materialized influences to estimate upper bounds and lower bounds of the influence spread of users so as to avoid computing the actual influences of users with small influences. To summarize, we make the following contributions.

- We propose a best-effort method that answers a TIM query online while keeping a high influence spread with an approximation ratio of $1-\frac{1}{e}$ under the MIA model.
- We devise effective techniques to estimate the upper bounds of the topic-aware influence of each user and utilize the upper bounds to prune a large number of users with small influences.
- We develop a faster topic-sample-based method to solve the topic-aware influence maximization problem with an approximation ratio $\epsilon \cdot (1-\frac{1}{e})$.
- Experimental results on real datasets show our methods significantly outperform state-of-the-art methods.

The rest of this paper is organized as follows. Section 2 formulates the problem. We devise a best-effort algorithm with $1-\frac{1}{e}$ approximation ratio in Section 3. Section 4 discusses bound-estimation techniques. A topic-materialization-based method with $\epsilon \cdot (1-\frac{1}{e})$ approximation ratio is proposed in Section 5. Section 6 reports the experiment results. We review related works in Section 7 and conclude in Section 8.

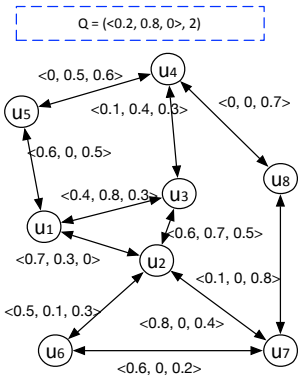
2. PRELIMINARIES

In this section, we first formally define the topic-aware influence maximization problem in Section 2.1. Then, we discuss a model for topic-aware influence computation in Section 2.2 and devise a greedy solution in Section 2.3.

2.1 Problem Formulation

Data Model. We model a social network as a topic-aware graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is a set of users and $\mathcal{E}$ contains directed edges connecting the users. Each edge $(u, v) \in \mathcal{E}$ is associated with a topic distribution $\langle pp^1_{u,v}, pp^2_{u,v}, \cdots, pp^Z_{u,v} \rangle$, where $pp^z_{u,v}$ is a weight on topic z (which denotes the activation probability that user v is activated by user u under topic z after u becomes active) and Z is the total number of topics. The topics reflect users' interests and the topic distribution on each edge can be computed by text-based topic discovery algorithms [1].

Query Model. We consider the topic-aware influence maximization (TIM) query $\mathcal{Q} = (\vec{\gamma}, k)$, where $\vec{\gamma} = \{\gamma^1, \gamma^2 \cdots, \gamma^Z\}$ is a query topic distribution in which γ^z is the probability on topic z, and k is an integer which denotes that the query asks for k users to be selected as seeds.



Edge	Topic 1	Topic 2	Topic 3
(1, 2)	0.7	0.3	0
(1, 3)	0.4	0.8	0.3
(1, 5)	0.6	0	0.5
(2, 3)	0.6	0.7	0.5
(2, 6)	0.5	0.1	0.3
(2, 7)	0.8	0	0.4
(3, 4)	0.1	0.4	0.3
(4, 5)	0	0.5	0.6
(4, 8)	0	0	0.7
(6, 7)	0.6	0	0.2
(7, 8)	0.1	0	0.8

Figure 1: A Running Example.

Topic-Aware Information Propagation. For each query, initially every user in $\mathcal{G}$ is *inactive*, and then k users are selected as *seeds* to become *active*. The active users then start to activate their neighbors through the edges. In this paper, we consider the activation is *topic-aware*, i.e., the propagation degrees that a user is activated by a neighbor are different across topics. Suppose u is selected as a seed, which attempts to activate its out-neighbors. Given a query topic distribution $\vec{\gamma}$, the activation probability for activating its out-neighbor v, denoted by $pp(u, v|\vec{\gamma})$, is computed as

$$pp(u, v|\vec{\gamma}) = \sum_{z=1}^{Z} \{pp^z_{u,v} \cdot \gamma^z\}. \quad (1)$$

In particular, each active seed u has only one chance to activate each of its neighbors. After that, u stays active and stops the activation. Moreover, if v is newly activated, v will further attempt to activate its neighbors. This process terminates when there is no new user activated. To evaluate the effect of the process, we introduce the *influence spread*. The influence spread of a seed set $\mathcal{S}$ with k seed users under topic distribution $\vec{\gamma}$, denoted by $\sigma(\mathcal{S}|\vec{\gamma})$, is defined as the *expected* number of active users after the propagation.

Topic-Aware Influence Maximization Problem. Given a topic-aware social graph and a TIM query, the problem finds a k-size set of seed users from the social graph that maximizes the influence spread of the seeds under the query topic distribution. Next we give a formal definition.

DEFINITION 1 (TOPIC-AWARE INFLUENCE MAXIMIZATION). *Given a topic-aware social graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and a TIM query $\mathcal{Q} = (\vec{\gamma}, k)$, it finds a seed set $\mathcal{S}^* = \arg\max_{\mathcal{S}} \sigma(\mathcal{S}|\vec{\gamma})$, where $\mathcal{S} \in \mathcal{V}, |\mathcal{S}| = k$.*

EXAMPLE 1. *Figure 1 shows a running example, consisting of a social graph and a table providing the topic distribution on each edge. For simplicity, each edge is bidirectional, and the topic distributions in the two directions are the same. The box with dotted style border contains the query $\mathcal{Q}$. Under this query $\mathcal{Q}$, the top-2 seed set is $\mathcal{S} = \{u_3, u_5\}$, as they can achieve the maximum influence spread.*

2.2 Influence Computation Model

An essential building block for evaluating a TIM query is to compute influence spread $\sigma(\mathcal{S} \mid \vec{\gamma})$ of a seed set $\mathcal{S}$ given a topic distribution $\vec{\gamma}$. However, this computation has been proved to be complex, as stated in the following lemma.

LEMMA 1. *[6] Given a topic distribution $\vec{\gamma}$, the computation of influence $\sigma(\mathcal{S}|\vec{\gamma})$ of a seed set $\mathcal{S}$ is #P-hard.*

To achieve better performance in computing the influence spread, we utilize a maximum influence arborescence (MIA) model [6] as approximation. Under the MIA model, user u activates user v *only* through the *maximum influence path* between them. Specifically, a path from u to v, denoted by $\mathcal{P}_{u,v} = \langle u = w_1, w_2, \ldots, w_m = v\rangle$, is a non-cyclic sequence of users, where adjacent users are connected by edges in $\mathcal{E}$. Through $\mathcal{P}_{u,v}$, u can activate v with the activation probability

$$pp(\mathcal{P}_{u,v}|\vec{\gamma}) = \prod_{k=1}^{m-1} pp(w_k, w_{k+1}|\vec{\gamma}). \quad (2)$$

As there may be multiple paths from u to v, the maximum influence path, denoted by $u \rightsquigarrow v$, is the one with the maximum activation probability among them, i.e.,

$$u \rightsquigarrow v = \arg\max_{\mathcal{P}_{u,v}} \{pp(\mathcal{P}_{u,v}|\vec{\gamma})\}. \quad (3)$$

If there are multiple paths with the maximum activation probability, we select any such path as $u \rightsquigarrow v$. MIA model also utilizes a threshold θ to remove *insignificant* maximum influence paths: if $pp(u \rightsquigarrow v|\vec{\gamma}) < \theta$, v is assumed not to be activated by u and we do not consider the maximum influence path from u to v. We set $\theta = 0.1$ in the examples in the remaining of the paper.

By assembling the paths from all users in $\mathcal{V}$ to a target user v, we can build a tree structure with v as its root and utilize this tree for influence computation. Specifically, given a topic distribution $\vec{\gamma}$ and a seed set $\mathcal{S}$, we compute activated probability $ap(v|\mathcal{S}, \vec{\gamma})$, which is the probability that v is activated by $\mathcal{S}$ under $\vec{\gamma}$ as follows. If v is already in the seed set $\mathcal{S}$, the probability can be computed trivially as $ap(v|\mathcal{S}, \vec{\gamma}) = 1$. Otherwise, we consider the child users $\mathcal{C}(v)$ of v in the tree. For each child node $w \in \mathcal{C}(v)$, its influence on v consists of both activated probability $ap(w|\mathcal{S}, \vec{\gamma})$ of $\mathcal{S}$ on w and activation probability $pp(w, v|\vec{\gamma})$ of w on v. To compute $ap(v|\mathcal{S}, \vec{\gamma})$, we combine influences of all the children to obtain the probability that v is influenced by *at least one* child in $\mathcal{C}(v)$. We first compute the probability of the complementary event, i.e., v cannot be influenced by any of its children, while assuming children's influences are independent to each other. Then, the activated probability $ap(v|\mathcal{S}, \vec{\gamma})$ is computed as

$$ap(v|\mathcal{S}, \vec{\gamma}) = 1 - \prod_{w \in \mathcal{C}(v)} \big(1 - ap(w|\mathcal{S}, \vec{\gamma}) \cdot pp(w, v|\vec{\gamma})\big), \quad (4)$$

where $v \notin S$ and, similarly to v, probability $ap(w|\mathcal{S}, \vec{\gamma})$ can be recursively computed from its children.

Overall, based on the MIA model, we compute the influence $\sigma(\mathcal{S}|\vec{\gamma})$ by summing the activated probabilities of all the users in the social graph, i.e.,

$$\sigma(\mathcal{S}|\vec{\gamma}) = \sum_{v \in \mathcal{V}} ap(v|\mathcal{S}, \vec{\gamma}) \quad (5)$$

2.3 Complexity and Greedy Algorithm

Complexity. Even with the approximate computation model MIA, the evaluation of a TIM query is still computationally complex. It is not difficult to prove that this problem is NP-hard by a reduction from the conventional influence maximization problem without topics [15, 6].

LEMMA 2. *Topic-aware influence maximization under the MIA influence computation model is NP-hard.*

PROOF. Given an instance of conventional influence maximization, we can always construct an instance of our problem with a space of $Z = 1$ topic, and the solution of our problem embeds a solution of the conventional problem. As the conventional problem is NP-hard, we prove the lemma. □

Despite the above lemma reveals that computing the best seed set is intractable in general, we show that the influence function $\sigma(\mathcal{S}|\vec{\gamma})$ has two properties that enable us to develop a good approximation algorithm. First, the influence function is monotonic, that is, for any $\mathcal{S}_1 \subseteq \mathcal{S}_2$, $\sigma(\mathcal{S}_1|\vec{\gamma}) \leq \sigma(\mathcal{S}_2|\vec{\gamma})$. Second, this function is submodular, that is, for any $\mathcal{S}_1 \subseteq \mathcal{S}_2$ and a user u, $\sigma(\mathcal{S}_1 \cup \{u\}|\vec{\gamma}) - \sigma(\mathcal{S}_1|\vec{\gamma}) \geq \sigma(\mathcal{S}_2 \cup \{u\}|\vec{\gamma}) - \sigma(\mathcal{S}_2|\vec{\gamma})$, which is also known as the property of diminishing returns. Based on these properties, a greedy algorithm can achieve an approximation ratio of $1 - \frac{1}{e}$.

Greedy Algorithm. Next, we describe details of the greedy algorithm. Given a TIM query $\mathcal{Q} = (\vec{\gamma}, k)$, the algorithm initially computes activation probability $pp(u, v|\vec{\gamma})$ for each edge (u, v) in $\mathcal{E}$, and builds a tree structure for each $v \in \mathcal{V}$ by assembling maximum influence paths $\{u \rightsquigarrow v\}$ from all users in $\mathcal{V}$. Next, it selects seeds in k iterations. In each iteration, the algorithm computes the *marginal influence* of every user in $\mathcal{V} - \mathcal{S}$, where the marginal influence of user u given the current seed set $\mathcal{S}$, denoted by $\Delta\sigma(u|\mathcal{S}, \vec{\gamma})$, is

$$\Delta\sigma(u|\mathcal{S}, \vec{\gamma}) = \sigma(\mathcal{S} \cup \{u\}|\vec{\gamma}) - \sigma(\mathcal{S}|\vec{\gamma}). \quad (6)$$

The algorithm selects the user u^* maximizing the marginal influence, i.e., $u^* = \arg\max_{u \in \mathcal{V}-S} \{\Delta\sigma(u|\mathcal{S}, \vec{\gamma})\}$, and inserts u^* into the seed set $\mathcal{S}$. Then, it updates the activated probability $ap(v|\mathcal{S}, \vec{\gamma})$ for each $v \in \mathcal{V}$ and continues to the next iteration. Finally, after k iterations, the algorithm outputs the set $\mathcal{S}$ as the selected seed users.

Limitations. The greedy algorithm has a limitation that it computes marginal influence for every user u in $\mathcal{V} - \mathcal{S}$ in each iteration. This computation is rather expensive due to the following two reasons. First, the algorithm needs to on-the-fly build maximum influence paths from u to all users in $\mathcal{V}$, which involves complicated computation including both edge activation probability computation and path selection with Equation (3). Second, even with the computed maximum influence paths, the algorithm still has to compute the influence of seed set $\mathcal{S} \cup \{u\}$ by traversing the tree structure of every target user v in $\mathcal{V}$ (Equations (4) and (5)). For many users, however, this computation could be avoided if the users have limited influence under query topic distribution $\vec{\gamma}$. Therefore, if we can reduce the unnecessary computation for these "insignificant" users, we may essentially improve the performance. To this end, we first propose a best-effort algorithm in Section 3 to avoid computing the actual influence and then develop a much faster algorithm to further reduce the computation cost in Section 5.

3. A BEST-EFFORT FRAMEWORK

To achieve instant performance in answering a TIM query, we introduce a best-effort framework. The basic idea is that given a TIM query we estimate the upper bound of the marginal influence of each user and preferentially compute exact marginal influence for the users with larger upper bounds, so as to prune the insignificant users. The pseudocode is illustrated in Algorithm 1. The algorithm takes as

input a topic-aware social graph $\mathcal{G}$, a TIM query $\mathcal{Q} = (\vec{\gamma}, k)$ and a threshold θ, and outputs a k-size seed set $\mathcal{S}$ with the maximum influence $\sigma(\mathcal{S}|\vec{\gamma})$ under topic distribution $\vec{\gamma}$.

The algorithm consists of offline indexing and online search. In the offline phase, for each user $u \in \mathcal{V}$, it estimates the upper bound of its influence to all users in $\mathcal{V}$, denoted by $\hat{\sigma}(u)$ (line 3), and inserts the user with the upper bound into a priority queue $\mathcal{L}$, where $\mathcal{L}$ is sorted in descending order of the upper bounds.

For online search, given a TIM query $\mathcal{Q} = (\vec{\gamma}, k)$, the algorithm employs a max-heap $\mathcal{H}$ to improve the performance. It also selects the seeds iteratively. However, the essential difference is that, in each iteration, instead of computing accurate influences for all users, our algorithm utilizes $\mathcal{H}$ to preferentially access the users with larger upper bounds. To this end, on the one hand, it progressively moves the *promising* users from the offline computed priority queue $\mathcal{L}$ to the max-heap $\mathcal{H}$ so as to prune the users with insignificant initial influences. On the other hand, for the users inside $\mathcal{H}$, it defers the expensive computation of their exact marginal influence by maintaining a state for each user, which has three possible values: 1) *initial*: the upper bound of u in this iteration is obtained from $\mathcal{H}$ in the last iteration or added from $\mathcal{L}$; 2) *bounded*: the upper bound of the marginal influence of u is estimated by our estimation algorithm; 3) *exact*: the exact marginal influence of u is computed based on Equation (6). Based on the state information, it selects seeds iteratively as follows.

In the i-th iteration, the algorithm first initializes the candidate seeds: setting the state of each user $u \in \mathcal{H}$ as *initial* (because the seed set $\mathcal{S}$ has been updated and the marginal influence of u is not accurate), and then finds the next seed by repeating the following steps.

(1) Adding possible candidates from $\mathcal{L}$. It examines if it is necessary to move more candidates from $\mathcal{L}$ to $\mathcal{H}$ (line 10): it repeatedly pops the front of $\mathcal{L}$ until the criterion $\mathcal{H}.top() > \mathcal{L}.front()$ meets, and inserts each user u popped from $\mathcal{L}$ with its upper bound, i.e., $\langle u, \hat{\sigma}(u)\rangle$, into $\mathcal{H}$.

(2) Finding a seed from $\mathcal{H}$. The algorithm pops the top user u with its upper bound from $\mathcal{H}$ (line 11) and considers the following three cases.

Case 1: the upper bound of u is not updated in this iteration ($u.\texttt{state} = initial$). In this case, it estimates the upper bound of u's marginal influence by calling EstMarginUB (line 13) and changes u's state to *bounded*. Then, it inserts u with its updated upper bound back into $\mathcal{H}$. More details of EstMarginUB will be described later.

Case 2: the upper bound of u is estimated ($u.\texttt{state} = bounded$). In this case, it computes the accurate marginal influence u given the previous seed set $\mathcal{S}$ by calling CalcMargin (line 17) and changes the state of u to *exact*. Then, it inserts u with its accurate marginal influence back into $\mathcal{H}$. More details of CalcMargin will be described later.

Case 3: marginal influence of u is computed ($u.\texttt{state} = exact$). In this case, it can guarantee that u is better than any user in $\mathcal{H}$ and $\mathcal{L}$. Thus, u can be safely inserted into $\mathcal{S}$ as a new seed and the activated probabilities of users in $\mathcal{V} - \mathcal{S}$ are updated. Then, the algorithm stops visits $\mathcal{H}$ in this iteration and continues to the next iteration.

After k iterations, the algorithm produces $\mathcal{S}$ as the result.

Since the influence function $\sigma(\mathcal{S}|\vec{\gamma})$ has two properties: monotonic and submodular, it is not hard to prove that our best-effort algorithm achieves an approximation ratio of $1 - \frac{1}{e}$, as formalized in Lemma 3.

Algorithm 1: BestEffort $(\mathcal{G}, \mathcal{Q}, \theta)$

Input: $\mathcal{G} = (\mathcal{V}, \mathcal{E})$: Social graph; $\mathcal{Q} = (\vec{\gamma}, k)$: A query; θ: A threshold
Output: $\mathcal{S}$: A k-size seed set

```
// Offline -- Indexing
1  Initialize an empty priority queue L ;
2  for each u in V do
3      σ̂(u) ← EstInfUB (u, G, θ) ;
4      Insert ⟨u, σ̂(u)⟩ into L ;
// Online -- Search
5  Initialize an empty heap H ;
6  Initialize an empty set S ;
7  for i ← 1 to k do
8      for each u in H do u.state ← initial ;
9      repeat
10         InsertCandidates (H, L) ;
11         u ← H.pop() ;
12         if u.state = initial then
13             Δ̂σ(u|S, γ⃗) ← EstMarginUB (u, G, θ, γ⃗) ;
14             u.state ← bounded;
15             Insert ⟨u, Δ̂σ(u|S, γ⃗)⟩ into H;
16         else if u.state = bounded then
17             Δσ(u|S, γ⃗) ← CalcMargin (u, G, θ, γ⃗, S) ;
18             u.state ← exact ;
19             Insert ⟨u, Δσ(u|S, γ⃗)⟩ into H;
20         else if u.state = exact then
21             S = S ∪ {u};
22             Update ap(v|S, γ⃗) for each v in V − S;
23             break;
24     until H = ∅ ;
25     return S;
```

Lemma 3. *Our best-effort algorithm achieves an approximation ratio of* $1 - \frac{1}{e}$.

Example 2. *Given the query $\mathcal{Q}$ in Figure 1, the algorithm first computes the offline upper bound of influence for each user. For example, the offline upper bound of u_3 is* 4.738. *It scans the users ordered by their upper bounds, and uses* EstInfUB *to estimate the influence under query $\mathcal{Q}$. For example, the influence of u_3 under $\mathcal{Q}$ is* 6.317. *It stops when $\mathcal{H}.top() > \mathcal{L}.front()$. After $\mathcal{H}$ is updated, it selects the vertex with the maximum influence in $\mathcal{H}$, and if the status is* bounded, *it invokes the exact method to calculate its actual influence. For example, the influence of u_3 is* 3.107. *Then it re-inserts $\langle u_3, 3.107\rangle$ into $\mathcal{H}$. This process stops when the top user in $\mathcal{H}$ has the status* exact. *It then selects the vertex with the maximum influence under $\mathcal{Q}$ as the next seed, i.e., vertex u_3 with influence* 3.107. *Iteratively the algorithm selects the next seed u_5, and returns seed set $\{u_3, u_5\}$.*

Next, we discuss more details of the upper bound estimation and marginal influence computation embedded in the algorithm mentioned above.

Upper Bound Estimation. It is rather challenging to estimate the initial upper bound in EstInfUB (line 3) and the marginal upper bound in EstMarginUB (line 13) of

Algorithm 1, because (1) the topic-aware influence is heavily sensitive to query topic distribution even for the same set of seeds; (2) it is hard to predict the query topic distribution; (3) it is space consuming to materialize the influence upper bound for each user and every possible topic distributions. To address these inherent challenges, we propose novel topic-aware bound estimation techniques in Section 4.

Online Marginal Influence Computation. We discuss how to on-the-fly compute the marginal influence $\Delta\sigma(u|\mathcal{S},\vec{\gamma})$ of u given a seed set $\mathcal{S}$ and a query distribution $\vec{\gamma}$. A straightforward method is to directly compute the increase of the activated probability $ap(v|\mathcal{S}\cup\{u\},\vec{\gamma}) - ap(v|\mathcal{S},\vec{\gamma})$ for every target user $v \in \mathcal{V}-\mathcal{S}$ and then sums the increases to obtain the marginal influence. Specifically, the method first computes all the maximum influence paths $\{u \rightsquigarrow v\}$ from u. Then, it has to visit each individual target user v again and computes $ap(v|\mathcal{S}\cup\{u\},\vec{\gamma}) - ap(v|\mathcal{S},\vec{\gamma})$ using Equation (4).

However this method is rather expensive as it computes many influence paths. To address this issue, we propose an alternative algorithm that simultaneously computes marginal influence paths and activated probability. The basic idea is similar to the single-source shortest path algorithm. The algorithm traverses the social graph $\mathcal{G}$ from u and updates the maximum influences of u on visited users and the corresponding paths. For each visited user v, it computes the increase of its activated probability by u through the current maximum influence path $u \rightsquigarrow v$. We use $v.\Delta_u$ to denote the increase in activated probability $ap(v|\mathcal{S}\cup\{u\},\vec{\gamma}) - ap(v|\mathcal{S},\vec{\gamma})$ of v caused by u. As it would incur large cost to compute $v.\Delta_u$ from scratch, we devise an *incremental* method that incrementally computes $v.\Delta_u$ from its previous user w in the maximum influence path $u \rightsquigarrow v$, as stated as follows.

LEMMA 4. *Suppose w is the preceding user to v in the maximum influence path $u \rightsquigarrow v$, and there exists an edge (w,v) in $\mathcal{E}$. We have:*

$$v.\Delta_u = w.\Delta_u \cdot pp(w,v|\vec{\gamma}) \cdot \Gamma, \tag{7}$$

where $\Gamma = \prod_{w' \in \mathcal{C}(v)/\{w\}} \{1 - ap(w|\mathcal{S},\vec{\gamma}) \cdot pp(w',v|\vec{\gamma})\}$.

PROOF. As w is included in maximum influence path $u \rightsquigarrow v$, according to Equation (4), we have $ap(v|\mathcal{S}\cup\{u\},\vec{\gamma}) = 1 - \big(1 - ap(w|\mathcal{S}\cup\{u\},\vec{\gamma}) \cdot pp(w,v|\vec{\gamma})\big) \cdot \Gamma$. Similarly, we also have $ap(v|\mathcal{S},\vec{\gamma}) = 1 - \big(1 - ap(w|\mathcal{S},\vec{\gamma}) \cdot pp(w,v|\vec{\gamma})\big) \cdot \Gamma$. Therefore, $v.\Delta_u = ap(v|\mathcal{S}\cup\{u\},\vec{\gamma}) - ap(v|\mathcal{S},\vec{\gamma}) = \big(ap(w|\mathcal{S}\cup\{u\},\vec{\gamma}) - ap(w|\mathcal{S},\vec{\gamma})\big) \cdot pp(w,v|\vec{\gamma}) \cdot \Gamma$. So we prove the lemma. □

The pseudocode of online marginal influence computation is illustrated in Algorithm 2. The algorithm maintains $v.\texttt{inf}$ and $v.\Delta_u$ for each user v, which respectively represent the influence ($pp(u \rightsquigarrow v|\vec{\gamma})$) on v and the increase of v's activated probability. Based on this, it utilizes a heap $\mathcal{M}$ to access the users in descending order of $v.\texttt{inf}$. At each time, the algorithm pops the top user w from $\mathcal{M}$ and considers the following two cases. If w has already been selected as a seed $w \in \mathcal{S}$ or the influence of u on w is insignificant ($w.\texttt{inf} < \theta$), it stops the traversal from w. Otherwise, it visits each unvisited edge (w,v) of w to update v: if v receives a larger influence $v.\texttt{inf}$ through the path including w, it accordingly updates $v.\Delta_u$ based on Equation (7). Finally, after traversing $\mathcal{G}$, the algorithm computes the marginal influence $\Delta\sigma(u|\mathcal{S},\vec{\gamma})$ of u as the summation of $v.\Delta_u$ for the users with significant influence ($v.\texttt{inf} \geq \theta$).

Algorithm 2: CALCMARGIN $(u,\mathcal{G},\theta,\vec{\gamma},\mathcal{S})$

Input: u: A user; $\mathcal{G}$: A social graph; θ: A threshold; $\vec{\gamma}$: A topic distribution; $\mathcal{S}$: A seed set
Output: $\Delta\sigma(u|\mathcal{S},\vec{\gamma})$: Marginal influence of u

```
1  Initialize a max-heap M ;
2  u.inf ← 1; u.Δ_u ← 1 − ap(u|S,γ⃗) ;
3  Insert ⟨u, u.inf⟩ into M ;
4  while M ≠ ∅ do
5      w ← M.pop() ;
6      if w ∈ S or w.inf < θ then continue;
7      for v ∈ C(w) do
8          if edge (w,v) is visited then continue ;
9          inf ← w.inf × pp(w,v|γ⃗) ;
10         if v ∉ M or inf > v.inf then
11             v.inf ← inf ;
12             v.Δ_u ← w.Δ_u × pp(w,v|γ⃗) ×
               ∏_{w'∈C(v)/{w}} {1 − ap(w'|S,γ⃗) · pp(w',v|γ⃗)}
               if v ∉ M then Insert ⟨v, v.inf⟩ into M ;
13             else Adjust M due to new v.inf;
14 Δσ(u|S,γ⃗) ← Σ_{v∈V, v.inf≥θ} {v.Δ} ;
15 return Δσ(u|S,γ⃗) ;
```

4. TOPIC-AWARE BOUND ESTIMATION

This section discusses bound estimation techniques to address the challenge mentioned in Section 3. We introduce three strategies for estimating upper bound of the initial influence $\sigma(u|\vec{\gamma})$ for each user u in Section 4.1. We discuss how to estimate the upper bound of the marginal influence $\Delta\sigma(u|\mathcal{S},\vec{\gamma})$ given an existing seed set $\mathcal{S}$ in Section 4.2.

4.1 Bound Estimation for Initial Influence

In the following, we discuss several strategies for estimating upper bounds of initial influences, which can be used in function ESTINFUB (line 3 of Algorithm 1).

Precomputation Based Estimation. A straightforward method is to pre-compute a query-independent upper bound for each user u in the offline component, which is then used for any online query. To this end, we derive a new graph $\mathcal{G}'$ from $\mathcal{G}$ by setting the activation probability of each edge (u,v) as the maximum probability across topics, i.e., $pp(u,v) = \max_{z=1}^{Z} pp^z_{u,v}$. On top of $\mathcal{G}'$, we derive the maximum influence path $u \rightsquigarrow' v$ from u to every user v, and compute the activation probability $pp(u \rightsquigarrow' v)$. Then, we estimate an upper-bound of u's initial influence as

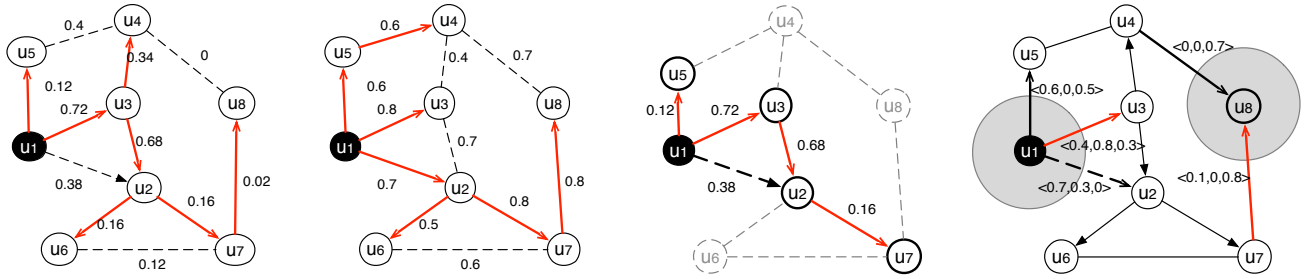
$$\hat{\sigma}_{\mathrm{P}}(u) = \sum_{v \in \mathcal{G}'} pp(u \rightsquigarrow' v). \tag{8}$$

We compute the upper bound $\hat{\sigma}_{\mathrm{P}}(u)$ offline. For an online query with topic distribution $\vec{\gamma}$, we compute an upper bound

$$\hat{\sigma}_{\mathrm{P}}(u|\vec{\gamma}) = \sum_{z=1}^{Z} \gamma^z \cdot \hat{\sigma}_{\mathrm{P}}(u) = \hat{\sigma}_{\mathrm{P}}(u). \tag{9}$$

It is easy to prove the derived $\hat{\sigma}_{\mathrm{P}}(u|\vec{\gamma})$ is indeed an upper bound, as $u \rightsquigarrow' v$ in $\mathcal{G}'$ has larger influence than any path from u to v in $\mathcal{G}$ under any $\vec{\gamma}$.

EXAMPLE 3. *Figure 2(b) illustrates this precomputation-based method to estimate an upper bound for u_1. In the offline component, graph $\mathcal{G}'$ is generated by considering the maximum probability in each edge and then the maximum*



(a) Exact Calculation (b) Precomputation based (c) Local-graph based (d) Neighborhood based

Figure 2: Upper Bound Estimation for The Initial Influence of User u_1.

influence path from u_1 to other users are computed (as indicated by red lines). For instance, the maximum influence path from u_1 to u_4 is $\langle u_1, u_5, u_4\rangle$ and the corresponding probability is $0.6 \times 0.6 = 0.36$. Similarly, we can compute the upper bound for other users. Overall, by summing such probabilities of all users, we obtain an upper-bound $\hat{\sigma}_{\mathrm{P}}(u_1) = 4.8$.

However, the precomputation-based upper bounds may be loose, as the actual activation probability $pp(u \rightsquigarrow v)$ given query $\vec{\gamma}$ may follow a different path and thus be much smaller than the pre-computed one $pp(u \rightsquigarrow' v)$. For example, given query $\vec{\gamma} = \langle 0.2, 0.8, 0.0\rangle$, the maximum influence path from u_1 to u_2 changes to $\langle u_1, u_3, u_2\rangle$ and the corresponding influence is $0.72 \times 0.68 = 0.49$, as shown in Figure 2(a). This actual influence is much smaller than the pre-computed 0.7, which is derived from the path $\langle u_1, u_2\rangle$.

Local Graph Based Estimation. The key problem of the precomputation based method is that maximum influence paths are quite *sensitive* to queries: they may dramatically change under different topic distributions. This means that the offline-computed bounds cannot be applied in all cases. A better estimation method is to (1) on-the-fly compute maximum influence paths for some users that may be largely affected by the query and (2) use the precomputation-based bounds for other users that are insignificantly affected by the query. To support the first case, on top of the graph $\mathcal{G}'$ derived in the precomputation-based method, for each user u, we identify each user v such that the influence of u on v is relatively large, i.e., $pp(u \rightsquigarrow' v) \geq \tau$, where τ is a parameter determined by experiments and usually set as $\sqrt{\theta}$. Then, we materialize a subgraph $\mathcal{G}_{\mathrm{L}}(u)$ of $\mathcal{G}$ consisting of the selected users and the edges among them. For the second case, for the user u, we also precompute its influence for users with influence not larger than $\frac{\theta}{\tau}$, i.e.,

$$\hat{\sigma}_{\mathrm{P}}(u|\frac{\theta}{\tau}) = \sum_{v \in \mathcal{G}'} pp(u \rightsquigarrow' v), \tag{10}$$

where $pp(u \rightsquigarrow' v) \leq \frac{\theta}{\tau}$. We materialize the local graph $\mathcal{G}_{\mathrm{L}}(u)$ and bound $\hat{\sigma}_{\mathrm{P}}(u|\frac{\theta}{\tau})$ offline and then utilize them to estimate tighter upper bounds online.

Given a query with a topic distribution $\vec{\gamma}$, we on-the-fly compute *topic-aware local maximum influence path* $\mathcal{P}_{u,v}$ from u to each user v in the local graph $\mathcal{G}_{\mathrm{L}}(u)$ based on Equation 2. It then considers the following two cases.

On the one hand, if $pp(\mathcal{P}_{u,v}|\vec{\gamma}) \geq \tau$, we can prove that the path is actually the maximum influence path $u \rightsquigarrow v$ under distribution $\vec{\gamma}$, as formally stated as follows. Thus, $pp(\mathcal{P}_{u,v}|\vec{\gamma}) \geq \tau$ is actual activation probability of u on v.

Lemma 5. *Consider the local maximum influence path $\mathcal{P}_{u,v}$ from u to v in $\mathcal{G}_{\mathrm{L}}(u)$. If $pp(\mathcal{P}_{u,v}|\vec{\gamma}) \geq \tau$, then $\mathcal{P}_{u,v}$ is the global maximum influence path $u \rightsquigarrow v$ in $\mathcal{G}$.*

On the other hand, if $pp(\mathcal{P}_{u,v}|\vec{\gamma}) < \tau$, as $\mathcal{P}_{u,v}$ may not be the global maximum influence path in $\mathcal{G}$, we simply use τ to estimate the upper bound of u's influence on v, i.e., $pp(\mathcal{P}_{u,v}|\vec{\gamma}) = \tau$.

For any user w outside the local graph, if $pp(\mathcal{P}_{u,w}|\vec{\gamma}) \geq \theta$, there must exist a user v in the local graph $\mathcal{G}_{\mathrm{L}}(u)$ such that $pp(u \rightsquigarrow' v) \geq \frac{\theta}{\tau}$. Thus we can use $pp(\mathcal{P}_{u,v}|\vec{\gamma}) \cdot \hat{\sigma}_{\mathrm{P}}(v|\frac{\theta}{\tau})$ to estimate the bounds for such users.

Thus the overall bound is computed as below.

$$\hat{\sigma}_{\mathrm{L}}(u|\vec{\gamma}) = \sum_{v \in \mathcal{G}_{\mathrm{L}}(u)} pp(\mathcal{P}_{u,v}|\vec{\gamma}) \cdot \hat{\sigma}_{\mathrm{P}}(v|\frac{\theta}{\tau}). \tag{11}$$

Example 4. *Figure 2(c) shows an example of local graph based estimation. When $\tau = 0.4$, the method materializes a local graph for u_1 consisting of u_2, u_3, u_5 and u_7. Given our example query, it computes the actual probability for each edge in the local graph and derives the local maximum influence paths. For example, the path for u_2 is $\langle u_1, u_3, u_2\rangle$ with probability $0.72 \times 0.68 = 0.46$. As this probability is larger than τ, we must have $pp(u_1 \rightsquigarrow u_2) = 0.46$. For another user u_5, as the probability of its local maximum influence path is 0.12, we use $\tau = 0.4$ to estimate its upper bound. Overall, the upper-bound estimate is 4.17, which is tighter than the estimate computed by the precomputation-based method.*

However, this method is still not satisfactory due to the following reasons. For the user v such that $v \in \mathcal{G}_{\mathrm{L}}(u)$ and $pp(\mathcal{P}_{u,v}|\vec{\gamma}) < \tau$, τ may be too loose as an upper bound. Moreover, the method cannot improve the estimation for the users outside $\mathcal{G}_{\mathrm{L}}(u)$. Although each of such users may not bring large estimation error as u has quite limited influence on that user, the number of these users is huge according to our experimental observations. As a result, the method may produce large errors due to the "long tail" effect. Moreover, as the method needs to on-the-fly compute local maximum influence, it would incur large estimation cost which may also affect the overall performance.

Neighborhood-Based Estimation. To address the limitations of the aforementioned approaches, we further introduce a *neighborhood*-based method. The intuition is that, given a query $\vec{\gamma}$, the maximum influence path $u \rightsquigarrow v$ from u to v must include a user w_1 in u's out-neighborhood and a user w_2 in v's in-neighborhood (sometimes, w_1 and w_2 may be the same user). As such, we can estimate the upper bound of $pp(u \rightsquigarrow v|\vec{\gamma})$ by considering w_1 and w_2 to reduce the estimation errors.

Formally, let $\mathcal{N}^o(u)$ denote the out-neighborhood set of user u, i.e., $\mathcal{N}^o(u) = \{w \mid (u,w) \in \mathcal{E}\}$, and $\mathcal{N}^i(u)$ denote the in-neighborhood set of user u, i.e., $\mathcal{N}^i(u) = \{w \mid (w,u) \in \mathcal{E}\}$. Given a query topic distribution $\vec{\gamma}$, let $\hat{pp^o}_u$ be the maximum influence in $\mathcal{N}^o(u)$, i.e., $\hat{pp^o_u} = \max_{w \in \mathcal{N}^o(u)} \{pp(u,w|\vec{\gamma})\}$.

Algorithm 3: CalcBoundLocal $(u, \mathcal{G}, \vec{\gamma}, \mathcal{S}, \theta, \tau)$

Input: u: User; $\mathcal{G}$: Social graph; $\vec{\gamma}$: Topic distribution; θ, τ: A threshold; $\mathcal{S}$: A seed set;
Output: $\hat{\Delta\sigma}(u|\mathcal{S}, \vec{\gamma})$: Upper bound of marginal influence of u

// Offline -- Dijkstra bound
for *each* u *in* $\mathcal{V}$ **do**
  $\hat{\sigma}_{\mathtt{P}}(u|\frac{\theta}{\tau}) = \sum_{v \in \mathcal{G}'} pp(u \rightsquigarrow' v)$;
  Calculate Local Graph $\mathcal{G}_{\mathtt{L}}(u)$;
// Online -- Local-graph
$\hat{\sigma}_{\mathtt{L}}(u|\vec{\gamma}) = \sum_{v \in \mathcal{G}_{\mathtt{L}}(u)} pp(\mathcal{P}_{u,v}|\vec{\gamma}) \cdot \hat{\sigma}_{\mathtt{P}}(v|\frac{\theta}{\tau})$;
$\Delta_u = 1 - ap(u|\mathcal{S}, \vec{\gamma})$;
$\hat{\Delta\sigma}(u|\mathcal{S}, \vec{\gamma}) = \Delta_u \cdot \hat{\sigma}(u)$;

User	Topics		
	1	2	3
1	0.7	0.8	0.6
2	0.8	0.7	0.5
3	0.6	0.8	0.5
4	0.1	0.4	0.7
...	...	...	...

(a) Estimating $\hat{pp}_u^z$

User	Topics		
	1	2	3
1	3.74	3.2	3.72
2	4.64	4.33	3.95
3	3.96	3.68	3.55
4	1.69	2.74	4.44
...	...	...	...

(b) Estimating U_u^z

Figure 3: Example of Neighborhood Estimation.

Similarly, we can compute $\hat{pp}_v^i = \max_{w \in \mathcal{N}^i(v)} \{pp(w, v|\vec{\gamma})\}$. Based on these, we estimate an upper bound of probability $pp(u \rightsquigarrow v|\vec{\gamma})$, denoted by $\hat{pp}_{u,v}$, as

$$\hat{pp}_{u,v} = \begin{cases} \max\{\hat{pp}_u^o, \hat{pp}_v^i\} & \text{if } (u,v) \in \mathcal{E} \\ \hat{pp}_u^o \cdot \hat{pp}_v^i & \text{if } (u,v) \notin \mathcal{E} \end{cases} \quad (12)$$

For example, consider u_1 and u_8 in Figure 2(d). Given our example query, we can compute $\hat{pp}_{u_1}^i = pp(u_1, u_3|\vec{\gamma}) = 0.72$ and $\hat{pp}_{u_8}^i = pp(u_7, u_8|\vec{\gamma}) = 0.02$. As u_1 and u_8 are not connected in graph $\mathcal{G}$, we estimate the upper bound as $\hat{pp}_{u_1,u_8} = 0.72 \times 0.02 = 0.01$, which is significantly tighter than 0.4 estimated by the previous local-graph based method.

However, although the above scheme can achieve tighter bounds, it may incur large estimation cost. Initially, it needs to on-the-fly compute the user with the largest influence in every neighborhood. Even worse, for each user u, it has to enumerate every influencee v of u to estimate $\hat{pp}_{u,v}$, which is obviously unacceptable for online queries. To reduce the estimation cost without sacrificing effectiveness, we introduce a lightweight estimation algorithm. The idea is that, for each user u, we only materialize a vector of Z bounds, $\langle U_u^1, U_u^2, \cdots, U_u^Z \rangle$, where U_u^z is the bound for topic z. Then, given online query $\vec{\gamma}$, we simply estimate the upper bound of u by computing the dot-product of this vector and $\vec{\gamma}$, i.e.,

$$\hat{\sigma}_{\mathtt{N}}(u|\vec{\gamma}) = \sum_{z=1}^{Z} \gamma^z \cdot U_u^z. \quad (13)$$

Next, we discuss how to derive an upper bound U_u^z for each topic z in the offline component. To this end, we first consider u's out-neighborhood $\mathcal{N}^o(u)$ and estimate an upper bound $\hat{pp}_u^{o\,z}$ as the maximum influence of u on a user in $\mathcal{N}^o(u)$ through topic z, i.e., $\hat{pp}_u^{o\,z} = \max_{w \in \mathcal{N}^o(u)} pp_{u,w}^z$. Similarly we can compute $\hat{pp}_v^{i\,z} = \max_{w \in \mathcal{N}^i(v)} pp_{w,v}^z$. Then, we estimate the upper bound of u's influence on v, denoted by $\hat{pp}_{u,v}^z$ based on Equation (12): if there exists $(u,v) \in \mathcal{E}$, we estimate $\hat{pp}_{u,v}^z = \max\{\hat{pp}_u^{o\,z}, \hat{pp}_v^{i\,z}\}$; otherwise, $\hat{pp}_{u,v}^z = \hat{pp}_u^{o\,z} \cdot \hat{pp}_v^{i\,z}$. Finally, we compute U_u^z as summation of bounds estimated above, i.e.,

$$U_u^z = \sum_{v \in \mathcal{V}} \hat{pp}_{u,v}^z. \quad (14)$$

It is not difficult to prove that U_u^z is an upper bound, as formally stated in Lemma 6.

Algorithm 4: CalcBoundNeighborhood $(u, \mathcal{G}, \mathcal{S}, \vec{\gamma})$

Input: u: A user; $\mathcal{G}$: A social graph; $\mathcal{S}$: A seed set; $\vec{\gamma}$: A topic distribution;
Output: $\hat{\Delta\sigma}(u|\mathcal{S}, \vec{\gamma})$: Upper bound of marginal influence of u

// Offline -- Dijkstra bound
for *each* u *in* $\mathcal{V}$ **do**
  calculate $U_u^z, \forall 1 \le z \le Z$;
// Online -- Neighborhood based
$\hat{\sigma_N}(u) = \sum_{z=1}^{Z} \gamma^z \cdot U_u^z$;
$\Delta_u = 1 - ap(u|\mathcal{S}, \vec{\gamma})$;
$\hat{\Delta\sigma}(u|\mathcal{S}, \vec{\gamma}) = \Delta_u \cdot \hat{\sigma}(u)$;

Lemma 6. *Given any query topic distribution* $\vec{\gamma}$*, we must have* $\sum_{z=1}^{Z} \gamma^z \cdot U_u^z \ge \sum_{v \in \mathcal{V}} \hat{pp}_{u,v}$.

Example 5. *Figure 3(a) provides the computed* $\hat{pp}_u^{o\,z}$ *and* $\hat{pp}_u^{i\,z}$ *for each user* u *in* $\mathcal{G}$*. For example, the bound of* u_1 *on topic* 1*, i.e.,* $\hat{pp}_{u_1}^{o\,z=1} = 0.7$*, is computed by the maximum of the probabilities* $pp^{z=1}(u_1, u_2) = 0.7$*,* $pp^{z=1}(u_1, u_3) = 0.4$ *and* $pp^{z=1}(u_1, u_5) = 0.6$ *in* $\mathcal{N}^o(u_1)$*. Then, we can compute bounds* $\hat{pp}_{u,v}^{z=1}$*, e.g.,* $\hat{pp}_{u_1,u_2}^{z=1} = \max\{0.7, 0.8\}$ *as* $(u_1, u_2) \in \mathcal{E}$ *and* $\hat{pp}_{u_1,u_4}^{z=1} = 0.7 \cdot 0.1$ *as* $(u_1, u_4) \notin \mathcal{E}$*. Finally, we obtain* U_u^z *for each user* u *under each topic* z*, as shown in Figure 3(b). Then, given our query* $(0.2, 0.8, 0)$*, the estimated upper bound of* u_1*'s initial influence is* 3.3*, which is much closer to the exact influence of* 2.7.

4.2 Bound Estimation for Marginal Influence

This section discusses how to estimate an upper bound of the marginal influence $\Delta\sigma(u|\mathcal{S}, \vec{\gamma})$ of u given a seed set $\mathcal{S}$. Compared with the bound estimation for initial influence, the difference is that the activated probability of u changes from $ap(u|\mathcal{S}, \vec{\gamma})$ to 1, instead of from 0 to 1, and thus the marginal influence of u would be smaller than the initial influence. On the other hand, the marginal influence of u may be correlated with the previously selected seed set $\mathcal{S}$, which is also known as *co-influence* [19]. In our example in Figure 2(a), given $\mathcal{S} = \{u_2\}$, the marginal influence of u_1 on u_3 is heavily correlated with the influence of u_2.

The basic idea is to consider the marginal influence of u on itself, denoted by $\Delta_u = 1 - ap(u|\mathcal{S}, \vec{\gamma})$, which reflects the increased activated probability of u. Moreover, we also consider the upper bound of initial influence of user u estimated by the aforementioned method, $\hat{\sigma}(u|\vec{\gamma})$. We have a nice property that an upper bound of $\Delta\sigma(u|\mathcal{S}, \vec{\gamma})$ can be computed as the multiplication of the above two factors, i.e.,

$$\hat{\Delta\sigma}(u|\mathcal{S}, \vec{\gamma}) = (1 - ap(u|\mathcal{S}, \vec{\gamma})) \cdot \hat{\sigma}(u|\vec{\gamma}), \quad (15)$$

as stated in the following lemma, where $\hat{\sigma}$ is any estimation method, e.g., $\hat{\sigma}_{\mathtt{P}}$, $\hat{\sigma}_{\mathtt{L}}$, and $\hat{\sigma}_{\mathtt{N}}$. The intuition of the lemma is that if we do not consider the co-influence of existing seeds in $\mathcal{S}$, we can obtain an upper bound of the marginal influence.

LEMMA 7. *Given query distribution $\vec{\gamma}$ and an existing seed set $\mathcal{S}$, let $\Delta_u = 1 - ap(u|\mathcal{S},\vec{\gamma})$, we must have*

$$\Delta\sigma(u|\mathcal{S},\vec{\gamma}) \leq \hat{\Delta\sigma}(u|\mathcal{S},\vec{\gamma}) = (1 - ap(u|\mathcal{S},\vec{\gamma})) \cdot \hat{\sigma}(u|\vec{\gamma}).$$

PROOF. This lemma can be proved based on Lemmas 4 and 6. We omit the details due to the limitation of space. □

5. A TOPIC-SAMPLE APPROACH

To further improve the performance, this section proposes a much faster approach. The basic idea is to prematurely terminate the process of the iterative seed selection in Algorithm 1 with a user-defined approximation ratio $\epsilon \cdot (1 - 1/e)$ where $\epsilon \in (0, 1]$. This approach can significantly improve the efficiency, while keeping a theoretical guarantee on the influence spread of the selected seeds.

Formally, given a query topic distribution $\vec{\gamma}$, suppose we have already selected i seeds in the set $\mathcal{S}^{(i)}$. Then, instead of directly computing the remaining seeds, we examine some pre-materialized $k - i$ users, denoted by $\mathcal{S}^{(k-i)}$, in the heap $\mathcal{H}$, and estimate the following two bounds: 1) the lower bound $\mathcal{B}_\mathcal{L}$ of the actual influence of the existing seeds $\mathcal{S}^{(i)}$ plus the upper bounds of the users in $\mathcal{S}^{(k-i)}$, and 2) the upper bound $\mathcal{B}$ of the influence of any k-size set of users under $\vec{\gamma}$. More details of estimating these two bounds will be described later. Obviously, if $\mathcal{B}_\mathcal{L} > \mathcal{B} * \epsilon$, we can safely terminate the algorithm and return $\mathcal{S}^{(i)} \cup \mathcal{S}^{(k-i)}$ as the selected seeds. It is not difficult to prove that the influence $\sigma(\mathcal{S}^{(i)} \cup \mathcal{S}^{(k-i)}|\vec{\gamma})$ of the selected seeds given $\vec{\gamma}$ must not be smaller than $\epsilon \cdot (1 - 1/e)$ multiplying the influence of the optimal seed set, which results in an approximation ratio of $\epsilon \cdot (1 - 1/e)$.

The essential challenge in this approach is to estimate the lower bound $\mathcal{B}_\mathcal{L}$ and the upper bound $\mathcal{B}$. Notice that $\mathcal{B}$ is different from the upper bound in the previous section: $\mathcal{B}$ is an upper bound of any k-size seed set while the upper bound in the previous section is only for an individual user. Obviously, if these bounds are tight enough, then we can achieve $\mathcal{B}_\mathcal{L} > \mathcal{B} * \epsilon$ in an early iteration (small number of iterations) and thus may produce a much faster solution for seed selection. A straightforward estimation method is to directly utilize the techniques introduced in Section 4: we respectively estimate $\mathcal{B}_\mathcal{L}$ as the actual influence of the selected seeds $\sigma(\mathcal{S}^{(i)}|\vec{\gamma})$ and $\mathcal{B}$ as $\sigma(\mathcal{S}^{(i)}|\vec{\gamma}) + \sum_{u \in \mathcal{S}^{(k-i)}} \hat{\Delta\sigma}(u|\mathcal{S},\vec{\gamma})$ where $\hat{\Delta\sigma}(u|\mathcal{S},\vec{\gamma})$ is an upper bound of u's marginal influence. However, this method is ineffective, because $\mathcal{B}_\mathcal{L} > \mathcal{B} * \epsilon$ may occur in a large iteration i. To provide a tighter bound estimation, we introduce a topic-sample-based strategy.

Topic-Sample-Based Bound Estimation. The idea of this estimation strategy is to materialize a sample of Z-dimensional topic distributions from the space of Z topics. In the offline component, we employ each of the materialized topic distributions to formulate a graph and compute the corresponding k-size seed set. For online estimation, given a query $\vec{\gamma}$, we exploit the materialized topic distributions as well as their computed seed sets to improve the estimation of bounds $\mathcal{B}_\mathcal{L}$ and $\mathcal{B}$.

Formally, let $\vec{p}$ denote a Z-dimensional topic distribution sampled from the topic space. Notice that, different from $\vec{\gamma}$, $\vec{p}$ is not necessarily a topic distribution, i.e., $\sum_{z=1}^{Z} p^z$ may not be 1. For example, a sampled topic distribution could be $(0.22, 0.83, 0)$ where $\sum_{z=1}^{Z} p^z$ is 1.05 instead of 1. Let $P = \{\vec{p}_1, \vec{p}_2, \ldots, \vec{p}_m\}$ denote a set of such sampled topic distributions. The set P is then used for bound estimation. In the following, we first assume that P has already been selected and here focus on discussing how to use P to estimate $\mathcal{B}_\mathcal{L}$ and $\mathcal{B}$. After that, we will discuss how these topic distributions in P are sampled from the topic space.

In the offline component, for each sampled topic distribution $\vec{p} \in P$, we use the best-effort framework to compute and materialize the selected seed set, denoted by $\mathcal{S}(\vec{p})$, and the influence of the seed set $\sigma(\mathcal{S}|\vec{p})$.

For online estimation, given a query topic distribution $\vec{\gamma}$, we first pick two topic distributions from the sampled set P. The first topic distribution, called the upper bound sample, denoted by $\vec{p}_U$, is *pareto*-optimal than $\vec{\gamma}$, that is, for any topic z from 1 to Z, $\gamma^z \leq p_U^z$. If there are multiple topic distributions satisfying the *pareto*-optimal requirement, we choose the sample topic distributions with the smallest KL-Divergence distance $d_{KL}(\vec{\gamma}, \vec{p}_U)$, where

$$d_{KL}(\vec{p}_U, \vec{\gamma}) = \sum_i \gamma^i \log(\frac{\gamma^i}{p_U^i}). \tag{16}$$

If there is no such topic distribution, we have to retreat to the best-effort framework in Section 3.

The second topic distribution, called the lower bound sample, denoted by $\vec{p}_L$, is *pareto*-inferior than $\vec{\gamma}$, that is, for any topic z from 1 to Z, $\gamma^z \geq p_L^z$.

It is not hard to prove that: the influence $\sigma(\mathcal{S}(\vec{p}_U)|\vec{p}_U)$ of seed set $\mathcal{S}(\vec{p}_U)$ under $\vec{p}_U$ is an upper bound of the influence $\sigma(\mathcal{S}|\vec{\gamma})$, while $\sigma(\mathcal{S}(\vec{p}_L)|\vec{p}_L)$ is a lower bound of $\sigma(\mathcal{S}|\vec{\gamma})$. Based on these two samples, we estimate $\mathcal{B}$ as the above influence $\sigma(\mathcal{S}(\vec{p}_U)|\vec{p}_U)$. On the other hand, estimating $\mathcal{B}_\mathcal{L}$ is more tricky. Initially, we utilize $\sigma(\mathcal{S}(\vec{p}_L)|\vec{p}_L)$ to estimate $\mathcal{B}_\mathcal{L}$. Then, if $\mathcal{B}_\mathcal{L} \geq \epsilon \cdot \mathcal{B}$, we can safely terminate the algorithm and return $\mathcal{S}(\vec{p}_L)$ as the selected seeds for the query topic distribution $\vec{\gamma}$. Otherwise, we continue to select seeds based on our best-effort framework in multiple iterations. At each iteration, if u is selected as a new seed, we will insert u into $\mathcal{S}(\vec{p}_L)$ to replace the existing user in $\mathcal{S}(\vec{p}_L)$ with the least marginal influence. Next, we on-the-fly update the lower bound $\mathcal{B}_\mathcal{L}$ of the updated $\mathcal{S}(\vec{p}_L)$ by directly computing the influence of $\mathcal{S}(\vec{p}_L)$ with a cutoff threshold larger than θ, and further check if $\mathcal{B}_\mathcal{L} \geq \epsilon \cdot \mathcal{B}$ is satisfied. In particular, as updating the lower bound needs to re-compute the influence, we do not execute this update in every iteration. Instead, we update the lower bound periodically with a certain step.

EXAMPLE 6. *Given the query $\mathcal{Q}$ with $\vec{\gamma} = \langle 0.2, 0.8, 0\rangle$ in Figure 1, suppose we have an offline upper bound sample $\langle 0.22, 0.83, 0\rangle$ and a lower bound sample $\langle 0.19, 0.78, 0\rangle$. The lower bound sample contains the seed set $\{u_3, u_5\}$, and the overall influence is* 4.175. *The upper bound sample has influence* 4.354. *As* 4.175>4.354∗80%, *we can return $\{u_3, u_5\}$.*

Selecting Sampled Topic Distributions. A key problem to support the aforementioned estimation is to select a set P of sampled topic distributions in the offline component. Intuitively, to make the estimated bounds tighter, the sampled topic distributions in P are required to be as *close* to a query topic distribution $\vec{\gamma}$ as possible. Formally, for a seed set $\mathcal{S}$, we want the difference of the influence under $\vec{\gamma}$ and the influence under at least one $\vec{p} \in P$ to be as small as possible, i.e., minimizing the following formula.

$$\sum_{\vec{\gamma}} min_{\vec{p} \in P}(\sigma(\mathcal{S}|\vec{\gamma}) - \sigma(\mathcal{S}|\vec{p})). \tag{17}$$

Algorithm 5: TopicSample $(\mathcal{G}, \mathcal{Q}, \theta)$

Input: $\mathcal{G} = (\mathcal{V}, \mathcal{E})$: Social graph;
$\mathcal{Q} = (\vec{\gamma}, k, \epsilon)$: A query; θ: A threshold
Output: $\mathcal{S}$: A k-size seed set
`// Offline -- Indexing`
Calculate P from `Log` ;
Calculate upper bound $\sigma(\mathcal{S}(\vec{p}_U)|\vec{p}_U)$ in P ;
Calculate lower bound $\sigma(\mathcal{S}(\vec{p}_L)|\vec{p}_L)$ and seed set $\mathcal{S}(\vec{p}_L)$ in P ;
`// Online -- Search`
Initialize an empty set $\mathcal{S}^{(0)}$;
find closest upper/lower bound sample $\vec{p}_U/\vec{p}_L$;
$\mathcal{B} = \sigma(\mathcal{S}(\vec{p}_U)$; $\mathcal{B}_{\mathcal{L}} = \sigma(\mathcal{S}(\vec{p}_L)$;
if $B_l > B_u \cdot \epsilon$ **then**
  return $\mathcal{S}(\vec{p}_L)$;
for $i \leftarrow 1$ **to** k **do**
  `// BestEffort returns one new seed`
  $u =$ BestEffort $(\mathcal{G}, \mathcal{Q}, \theta)$;
  $\mathcal{S}^{(i)} = \mathcal{S}^{(i-1)} \cup \{u\}$;
  if $u \in \mathcal{S}(\vec{p}_L)$ **then**
    $\mathcal{S}(\vec{p}_L) = \mathcal{S}(\vec{p}_L)/u$;
  else
    $\mathcal{S}(\vec{p}_L) = \mathcal{S}(\vec{p}_L)/\arg\min\{v|\Delta\sigma(v|\mathcal{S}^{(i)}, \vec{\gamma}), v \in \vec{p}_L\}$;
  $\mathcal{B}_{\mathcal{L}} = \sigma(\mathcal{S}(\vec{p}_L) \cup \mathcal{S}^{(i)}|\vec{\gamma})$;
  if $B_l > B_u \cdot \epsilon$ **then**
    return $\mathcal{S}(\vec{p}_L) \cup \mathcal{S}^{(i)}$;
return $\mathcal{S}^{(k)}$;

However, the above formula is quite challenging to be minimized due to the following reasons. First, the formula assumes that every query $\vec{\gamma}$ is known in the offline component, which apparently does not hold in practice. To address this challenge, we employ a query log based solution that utilizes a set of existing topic distributions to simulate the possible distribution of $\vec{\gamma}$. This query log can be derived by the item-purchasing log considered in existing topic-aware influence studies [4, 1]. For example, in a movie rating website, the log consists of a set of movies, for which users have purchased tickets. Given this query log, we can use clustering algorithms to categorize topic distributions in the query log into m clusters, and then take the central point of each cluster as the sampled topic distribution. To this end, we can use K-means algorithms for implementing the clustering strategy. However, another challenge is that the formula in Equation (17) requires heavy calculation of $\sigma(\mathcal{S}|\vec{\gamma})$ in each iteration of the K-means algorithm, which may be very time consuming. To address this issue, we use a close approximate function as follows. The basic idea is to minimize the sum of KL-Divergence distance from sampled topic distributions to the distributions in query log, i.e., selecting the sample set P^* such that,

$$P^* = \arg\min \sum_{\vec{p} \in P} \sum_{\vec{s} \in P_i} d_{KL}(\vec{p}, \vec{s}), \tag{18}$$

where P_i is the subset of the query log in which each topic distribution $\vec{s} \in P_i$ has the smaller distance to the i-th sample in P than other samples.

The pseudo-code of the topic-sample-based algorithm is shown in Algorithm 5.

Table 1: Datasets.

Datasets	#Vertices	#Edges	AvgD	MaxD	Topic
`LiveJournal`	4847571	68993773	14.23	943	10
`DBLP`	944679	5255878	5.56	943	9
`Flixster`	31927	448320	14.04	331	10
`DIGGS`	14985	390160	26.01	3661	20

6. EXPERIMENTAL STUDY

We conducted extensive experiments to evaluate our proposed methods and compared our methods with the state-of-the-art approaches on efficiency and influence spread.

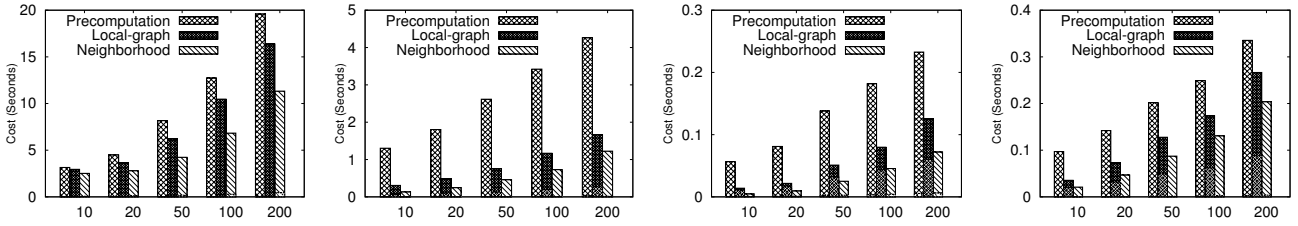
Datasets. We used the following four real-world datasets. `DIGGS` is an open social news dataset[1]. `DIGGS` contains an action log, which records users' activities of rating news stories. The propagation traces of the news stories were used by our `TopicSample` algorithm as the query log. `Flixster` is an American social movie site[2]. `Flixster` also contains an action log of users' activities of rating movies. `DBLP` is a DBLP co-author graph, which was downloaded from the online academic search service[3]. `LiveJournal` is a free on-line community called LiveJournal with almost 5 million members[4], which allows members to maintain journals and blogs. We adopted the TIC model proposed in [1] to compute topic distribution in each edge for the first two datasets. Since `DBLP` did not have an action log, we used research fields as topics to classify the conferences and categorized the authors and their social links by their related conferences. We also generated a random set of topic distributions as the query log. Similarly, as `LiveJournal` did not have an action log, we had to simulate topic distributions randomly, and also generated a random set of topic distributions as the query log. The four datasets are directed graphs and more details are shown in Table 1, where AvgD (MaxD) is the average (maximum) degree (all edges are bidirectional).

Algorithms. We implemented our proposed methods and compared with three state-of-the-art algorithms `PMIA` [6], `Greedy` [15], `INFLEX` [4] and `MIS` [5]. As `PMIA` did not consider topics, we extended `PMIA` to support our topic-aware problem as follows. To answer a query $\mathcal{Q} = (\vec{\gamma}, k)$, we first generated a graph $\mathcal{G}'$ by computing $pp(u,v) = \sum_{z=1}^{Z} \{pp^z_{u,v} \cdot \gamma^z\}$ for each edge (u, v), and then used the `PMIA` algorithm to select seeds. Similarly, `Greedy` also constructed a same graph and utilized the greedy algorithm [15] for seed selection. We obtained the source codes of `PMIA` and `MIS` from the authors. We implemented `INFLEX` by ourselves as we could not obtain the source code from the authors [4]. In all the experiments, θ and τ were set to 0.001 and $\sqrt{\theta}$ respectively.

Experiment settings. All the algorithms were implemented in C++ and compiled using GCC 4.7 with the -O3 flag. All the experiments were conducted on a computer running 64bit Ubuntu 14.04 with Intel Xeon E2650 2.0GHz processor and 16GB RAM.

Index Size and Time. Due to space constraints, we only report the index size and preprocessing time on the `DBLP` dataset, as shown in Table 2, with the configuration that $\theta = 0.001$, the number of samples is 800 and $k = 100$. We can see that (1) `PMIA` did not use any index; (2) `BestEffort` and `TopicSample` utilized indexes for upper bound estimations; (3) `TopicSample`, `INFLEX` and `MIS` also used samples.

[1] http://www.cs.sfu.ca/∼sja25/personal/datasets/
[2] http://www.isi.edu/ lerman/downloads/digg2009.html
[3] http://arnetminer.org
[4] http://snap.stanford.edu/data/soc-LiveJournal1.html



(a) LiveJournal ($Z = 10$) (b) DBLP ($Z = 9$) (c) Flixster ($Z = 10$) (d) DIGGS ($Z = 20$)

Figure 4: Evaluating Upper-Bound Estimation Algorithms ($\theta = 0.001$, Bottom bar: Estimation Cost, Top Bar: Accurate Influence Computation Cost).

Table 2: Index Size&Time(DBLP,800 samples,k=100.)

Methods	Data	Index	Sample	RT	Total	Time(s)
PMIA	337M	–	–	986M	1.32G	0
BestEffort	337M	87M	–	1.05G	1.39G	7
TopicSample	337M	87M	528K	1.05G	1.39G	1200
INFLEX	337M	–	600K	1.06G	1.40G	1400
MIS	337M	–	140K	986M	1.32G	1600

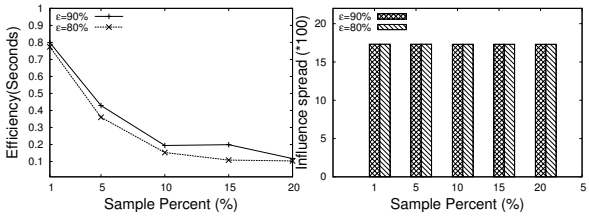
In addition to these indexes, the memory usage of each approach also included the dataset and runtime usage (RT). For MIS, we set the number of the precomputed landmarks to 10. The preprocessing time of TopicSample and INFLEX included the time for generating samples, and that of MIS included running PMIA on each landmark.

6.1 Evaluating Bound Estimation Methods

We compared our upper bound estimation techniques, Precomputation, LocalGraph, and Neighborhood, as discussed in Section 4. Figure 4 shows the results and we had the following observations. First, the precomputation-based method achieved the worst performance due to the rather loose upper bounds estimated. These loose bounds resulted in a large amount of time to compute the actual influence, as shown in the figures: the estimation only took a small amount of the total time and the time of actual influence computation took a much larger proportion. Second, LocalGraph achieved much better performance than Precomputation, because it used local graphs to estimate tighter bounds. Third, Neighborhood achieved the best performance, as Neighborhood estimated a much tighter upper bound and avoided many unnecessary actual influence computations, and thus the estimation cost was very low.

6.2 Evaluating Topic-Sample-Based Method

We compared the performance of TopicSample with different numbers of samples. We varied percentage of the topic distributions in the query log that were used to generate lower/upper bound samples (1%, 5%, 10%, 15%, 20%), and compared the efficiency of answering queries with $k = 100$ on the DBLP dataset. As shown in Figure 5, when the number of sample distributions was small (1%), the lower/upper bound samples had limited effect, and thus the BestEffort method was invoked for many times (and prematurely terminated later). As the number of sample distributions increased, the chance that the query found tight enough upper/lower bound distributions also increased, and thus the efficiency was improved as the BestEffort method was invoked infrequently (or prematurely terminated earlier). In addition, for different sample sizes, the influence spread was nearly the same. This is because, for a small sample size, TopicSample retreated to BestEffort and thus still got high influence spread. For a large sample size, TopicSample had a larger chance to find samples for estimation and thus also achieved high influence spread. As a large sample size had additional space overhead, we utilized the space budget to determine the sample size.



(a) Varying Samples (b) Varying Samples

Figure 5: Varying Sample ($\theta = 0.001$, $k = 100$).

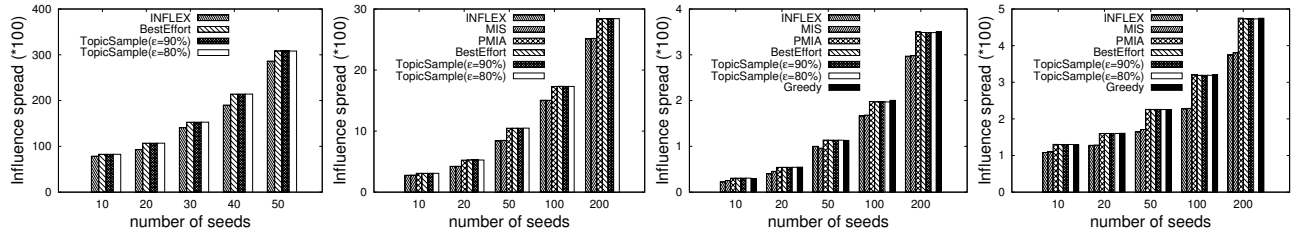
6.3 Comparison with Existing Approaches

6.3.1 Comparison on Influence Spread

We compared influence spread of the approaches, as shown in Figure 6. Note that PMIA and the precomputation step of MIS ran out of memory on the largest dataset LiveJournal, and thus their results are not provided. As Greedy took long time on large datasets, we only reported its results on DIGGS and Flixster. INFLEX and MIS had the worst influence spread on all the datasets. For instance, the influence spread of INFLEX and MIS on DIGGS was about 20% smaller that those of other approaches at each k, as shown in Figures 6(c)-6(d). The reason is due to their precomputation-based strategy: they identified precomputed topic distributions, which were most similar to the query, and then aggregated these precomputed information to generate seed sets. Such a strategy had no theoretical guarantee on influence spread, and may have weak performance when queries deviated from the precomputed topic distributions. In contrast, both PMIA and our BestEffort achieved larger influence spread, which was very similar to the influence spread of Greedy, because they had the theoretical guarantee that the influence spread of the selected seeds would not be smaller than $1 - 1/e$ times that of the optimal solution (under the MIA model for influence computation). Moreover, we also had an interesting observation that TopicSample achieved nearly the same influence spreads as PMIA, BestEffort and Greedy in practice, although it only had $80\% \cdot (1-1/e)$ or $90\% \cdot (1-1/e)$ approximation ratios. This is attributed to the "two-pronged" strategy of TopicSample: different from INFLEX, TopicSample selected high-influence users in the precomputed lower bound samples when the corresponding lower bound was tight enough; otherwise, it retreated to BestEffort to guarantee a theoretical influence spread.

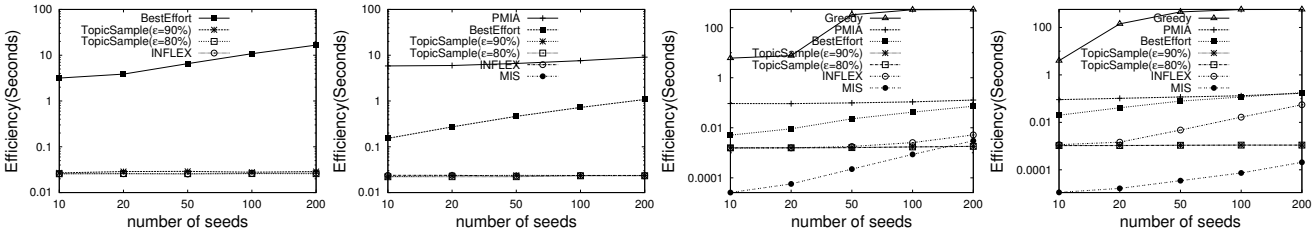
6.3.2 Comparison on Efficiency

We compared the approaches on efficiency. Similar to the previous section, PMIA and the preprocessing step of MIS ran out of memory on LiveJournal, and their results could not be shown. As shown in Figure 7, Greedy took a much larger amount of time even when k was small on a small dataset, because it involved many simulations for each candidate seed. PMIA took longer time than BestEffort,



(a) LiveJournal ($Z = 10$) (b) DBLP ($Z = 9$) (c) Flixster ($Z = 10$) (d) DIGGS ($Z = 20$)

Figure 6: Influence Spread Comparison ($\theta = 0.001$. MIS, PMIA ran out of memory on LiveJournal).



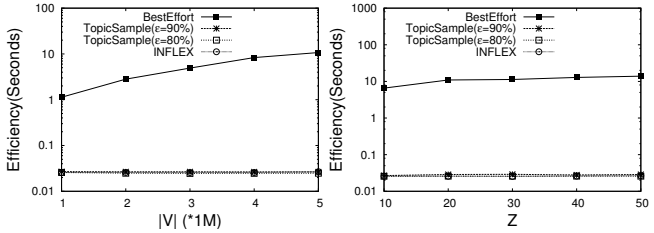
(a) LiveJournal ($Z = 10$) (b) DBLP ($Z = 9$) (c) Flixster ($Z = 10$) (d) DIGGS ($Z = 20$)

Figure 7: Efficiency Comparison ($\theta = 0.001$. MIS, PMIA ran out of memory on LiveJournal).

TopicSample, INFLEX, and MIS, because PMIA had to generate a new graph by computing propagation probability of each edge and compute an influence in-arborescence of each vertex, which was obviously not practical for online queries. Our proposed approach BestEffort outperformed PMIA by significant margins, and, with the increase of k, the time used by PMIA increased faster than that of BestEffort in most of the cases. The main reason is that BestEffort only calculated the actual influences of the users with large upper bounds and thus could prune many users with insignificant influences. Moreover, TopicSample achieved better performance on efficiency and selected seeds almost instantly due to the following reason. The lower bound and upper bound samples selected by our techniques proposed in Section 5 were tight and could terminate prematurely once the ϵ approximation requirement was satisfied. We also observed that INFLEX and MIS also achieved good performance on efficiency. This is mainly due to their lightweight online seed selection that simply selected some precomputed distributions and then aggregated their seed sets to produce k seeds. However, this lightweight strategy of INFLEX and MIS may lead to unsatisfactory influence spreads (much worse than TopicSample), as discussed in Section 6.3.1.

6.3.3 Scalability

We evaluated the scalability of the approaches, BestEffort, TopicSample and INFLEX on the largest LiveJournal dataset. Since the preprocessing part of MIS and PMIA ran out of memory and Greedy took very long time on the LiveJournal dataset, we did not report their results. We first varied the number of users in the social graph, i.e., $|\mathcal{V}| = 1, 2, 3, 4, 5$ millions. As shown in Figure 8(a), with the increase of $|\mathcal{V}|$, the elapsed time of BestEffort increased accordingly, as BestEffort needed to estimate bounds for more users and used more efforts to calculate actual influences for each user. However, the increase was sub-linear since the bounds estimated by BestEffort were tight, so as to prune large numbers of insignificant users. The elapsed time of TopicSample, INFLEX and MIS remained almost the same at each $|\mathcal{V}|$, since these approaches mainly relied on the precomputed distributions to answer a query, which were not much affected by the size of the social graph. In fact, TopicSample, MIS and INFLEX were only sensitive to the number of samples (or landmarks in the terminology of MIS).



(a) Varying $|\mathcal{V}|$ (b) Varying Z

Figure 8: Scalability on LiveJournal ($\theta = 0.001$).

We then compared the methods by varying the number of topics ($Z = 10, 20, 30, 40, 50$). As shown in Figure 8(b), with increasing Z, the elapsed time of BestEffort increased slowly. This is because the techniques affected by Z were propagation probability calculation in Equation (1) and bound estimation in Equation (13), which took a small percentage in the whole process of seed section and thus different numbers of topics did not significantly affect the overall performance. TopicSample, INFLEX, and MIS achieved similar performance at each value of Z due to the similar reason as before that they utilized the precomputed distributions to select seeds and these seed selection processes were not affected by the number of topics. Thus our method scaled well in terms of number of users and number of topics.

7. RELATED WORKS

Topic-aware Influence Maximization. Barbieri et al. [1] proposed the Topic-Aware Influence Cascade (TIC) model, within which the relationship strength between two vertices was computed by their topic preference learned from history activities on a social network. Based on the TIC model, Barbieri et al. [4] proposed a similarity-based method, INFLEX, to support topic-aware influence maximization. They used the greedy framework to offline calculate the result for some given topics based a query workload, and in the online process, they identified the most similar indexed topics and aggregated their precomputed seeds to answer the query. If there are no similar indexed topics, their influence spread may not be satisfactory. Chen et al. [5] introduced a preprocessing based strategy, MIS, for topic-aware influence maximization. They offline pre-computed some seed sets for

each topic, which were then used for online seed selection by aggregating these materialized topics. Our work is different from INFLEX and MIS in that (1) INFLEX and MIS have no influence-spread guarantee while our method has an influence-spread guarantee while keeping high performance. MIS has an influence-spread guarantee on some special kinds of datasets, such as *sub-additive* influence-spread functions and *topically separated* networks, while our method has an influence-spread guarantee on any types of datasets; (2) INFLEX, MIS, and our method use some materialized samples to improve the performance. However the strategies and objectives of generating the samples are different. MIS uses the dataset to generate the samples while INFLEX and our method utilize a query workload to generate the samples. INFLEX and MIS use the samples to directly answer a TIM query while our method utilizes the materialized influence to estimate upper and lower bounds so as to improve the performance with an influence-spread guarantee.

Influence Maximization in Social Networks. Domingos et al. proposed the influence maximization problem in a social graph [20, 11]. However, their proposed methods are probabilistic and have no guarantee on the influence spread. Kempe et al. [15] proposed two widely accepted discrete influence propagation models, Independent Cascade (IC) model and Linear Thresholds model, proved the influence maximization problem is NP-hard and proposed a greedy framework with $1 - \frac{1}{e}$ approximation ratio guarantee. However, their solution is rather inefficient and takes several hours on a medium dataset. There are many studies aiming at improving the performance of the greedy framework [18, 6, 12, 13]. Kimura et al. [17] used the shortest paths to approximate the actual spread process. Leskovec et al. [18] proposed a "lazy-forward" approach. Chen et al. [6] proposed the PMIA algorithm. The main idea is to use a local maximum influence out-arborescence to approximate a vertex's global influence. The arborescence is a tree structure composed of only shortest paths starting from the root, and the relationship graph is thus simplified. The similar idea is then applied to support the linear threshold model in [8]. Chen et al. [7] proposed degree-discount heuristics for an IC model where all propagation probabilities are the same. In [9], a community structure was utilized to aggregate the vertices with similar features to reduce the number of vertices needed to check. Kim et al. [16] proposed a parallelized approach with OpenMP meta-programming expressions. Jung et al. [14] approximated the real influence with linear equations. Borgs et al. [3] provided a fast algorithm to maximize social influence in nearly optimal time. Tang et al. [21] proposed an algorithm with near-optimal time complexity and novel heuristics for improving empirical efficiency. Cheng et al. [10] introduced interesting heuristics based on self-consistent ranking. Different from these approaches, we take into consideration topic information and study online topic-aware influence maximization.

8. CONCLUSION

We have studied the topic-aware influence maximization problem. We utilized a maximum influence arborescence (MIA) model to approximate the computation of influence spread. We proposed a best-effort algorithm with an approximation ratio of $1 - \frac{1}{e}$ under the MIA model. We devised precomputation-based and neighborhood-based techniques to estimate the upper bounds of location-aware influence spread and utilized the bunds to prune large numbers of users with small influences. We devised a faster topic-materialization-based algorithm with an approximation ratio of $\epsilon \cdot (1 - \frac{1}{e})$, which materialized the topic-aware influences of some topic distributions, utilized the materialized information to estimate the upper bounds and lower bounds of influence spreads and avoided computing actual influences of users with small influences based on the bounds. Experimental results on real-world datasets showed that our method outperformed the state-of-the-art approaches.

Acknowledgement. This work was partly supported by the 973 Program of China (2015CB358700 and 2011CB302206), and the NSFC project (61373024 and 61422205), YETP0105, Tencent, Huawei, SAP, the "NExT Research Center" (WBS: R-252-300-001-490), and the FDCT/106/2012/A3.

9. REFERENCES

[1] N. Barbieri, F. Bonchi, and G. Manco. Topic-aware social influence propagation models. In *ICDM*, pages 81–90, 2012.
[2] F. Bonchi. Influence propagation in social networks: A data mining perspective. *IEEE Intelligent Informatics Bulletin*, 12(1):8–16, 2011.
[3] C. Borgs, M. Brautbar, J. T. Chayes, and B. Lucier. Maximizing social influence in nearly optimal time. In *SODA*, pages 946–957, 2014.
[4] Çigdem Aslay, N. Barbieri, F. Bonchi, and R. A. Baeza-Yates. Online topic-aware influence maximization queries. In *EDBT*, pages 295–306, 2014.
[5] W. Chen, T. Lin, and C. Yang. Efficient topic-aware influence maximization using preprocessing. *CoRR*, abs/1403.0057, 2014.
[6] W. Chen, C. Wang, and Y. Wang. Scalable influence maximization for prevalent viral marketing in large-scale social networks. In *KDD*, pages 1029–1038, 2010.
[7] W. Chen, Y. Wang, and S. Yang. Efficient influence maximization in social networks. In *KDD*, pages 199–208, 2009.
[8] W. Chen, Y. Yuan, and L. Zhang. Scalable influence maximization in social networks under the linear threshold model. In *ICDM*, pages 88–97, 2010.
[9] Y.-C. Chen, W.-C. Peng, and S.-Y. Lee. Efficient algorithms for influence maximization in social networks. *Knowl. Inf. Syst.*, 33(3):577–601, 2012.
[10] S. Cheng, H. Shen, J. Huang, W. Chen, and X. Cheng. Imrank: influence maximization via finding self-consistent ranking. In *SIGIR*, pages 475–484, 2014.
[11] P. Domingos and M. Richardson. Mining the network value of customers. In *KDD*, pages 57–66, 2001.
[12] A. Goyal, F. Bonchi, and L. V. S. Lakshmanan. A data-based approach to social influence maximization. *PVLDB*, 5(1):73–84, 2011.
[13] A. Goyal, W. Lu, and L. V. S. Lakshmanan. Celf++: optimizing the greedy algorithm for influence maximization in social networks. In *WWW (Companion Volume)*, pages 47–48, 2011.
[14] K. Jung, W. Heo, and W. Chen. Irie: Scalable and robust influence maximization in social networks. In *ICDM*, pages 918–923, 2012.
[15] D. Kempe, J. M. Kleinberg, and É. Tardos. Maximizing the spread of influence through a social network. In *KDD*, pages 137–146, 2003.
[16] J. Kim, S.-K. Kim, and H. Yu. Scalable and parallelizable processing of influence maximization for large-scale social networks? In *ICDE*, pages 266–277, 2013.
[17] M. Kimura and K. Saito. Tractable models for information diffusion in social networks. In *PKDD*, pages 259–271, 2006.
[18] J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. M. VanBriesen, and N. S. Glance. Cost-effective outbreak detection in networks. In *KDD*, pages 420–429, 2007.
[19] G. Li, S. Chen, J. Feng, K.-L. Tan, and W.-S. Li. Efficient location-aware influence maximization. In *SIGMOD Conference*, pages 87–98, 2014.
[20] M. Richardson and P. Domingos. Mining knowledge-sharing sites for viral marketing. In *KDD*, pages 61–70, 2002.
[21] Y. Tang, X. Xiao, and Y. Shi. Influence maximization: near-optimal time complexity meets practical efficiency. In *SIGMOD*, pages 75–86, 2014.