Discovering Your Selling Points: Personalized Social Influential Tags Exploration

Yuchen Li [#], Ju Fan [*], Dongxiang Zhang [+], Kian-Lee Tan [#]
[#] *NUS* [*] *Renmin University of China* [+] *UESTC*
[#] {liyuchen, tankl}@comp.nus.edu.sg; [*] fanj@ruc.edu.cn; [+] zhangdo@uestc.edu.cn

ABSTRACT

Social influence has attracted significant attention owing to the prevalence of social networks (SNs). In this paper, we study a new social influence problem, called *personalized social influential tags exploration* (PITEX), to help any user in the SN explore how she influences the network. Given a target user, it finds a size-k tag set that maximizes this user's social influence. We prove the problem is NP-hard to be approximated within any constant ratio. To solve it, we introduce a sampling-based framework, which has an approximation ratio of $\frac{1-\epsilon}{1+\epsilon}$ with high probabilistic guarantee. To speedup the computation, we devise more efficient sampling techniques and propose best-effort exploration to quickly prune tag sets with small influence. To further enable instant exploration, we devise a novel index structure and develop effective pruning and materialization techniques. Experimental results on real large-scale datasets validate our theoretical findings and show high performances of our proposed methods.

1. INTRODUCTION

The prevalent of social networks (SNs) have boosted research on *social influence* due to its indispensable value in many applications, such as viral marketing and rumor control. Existing works on social influence analysis can be broadly classified into two categories. The first category devises models to capture how information is propagated in SNs. Examples include the classic independent cascade (IC) and linear threshold (LT) models [7, 19, 14], the triggering model [19, 36] in a more general form, and the recently proposed topic-aware models [2, 16, 26, 6]. The second category studies a key algorithmic problem, called *influence maximization (IM)*, that finds a set of k users to maximize the expected influence among all the users in an SN given a specific propagation model [6, 28, 35, 36].

Recently, the wide adoption of SNs has brought a new demand on *personalized social influence exploration* for *individual users* in the SNs. Unlike IM, which selects a set of users to maximize the network influence, it aims to help a *target* user explore *how she influences the network*. For example, political campaigns have an urgent demand to take advantage of SNs as important channels for opinion poll to analyze their "selling points". Fig. 1 shows an exemplary re-tweet network. The tweets containing the candidates' political standpoints, such as hashtags `foreign policy` and `social security`, are propagated through their followers to the entire SN. It is highly desirable for any candidate, say `Hilary Clinton`, to evaluate the "effect" of her standpoints and judge which ones can influence more people, e.g. `infrastructure rebuild` and `US-China relation`. To win more voters, her publicity manager should then spend more time on these issues for Hilary's subsequent public speeches or tweets. Another application is social media marketing, where businesses also want to position their marketing strategies by identifying the influential product features (e.g., `high-tech`, `energy-saving`) which are propagated to more people in SNs. Similarly, researchers want to explore their most influential research contributions from academia SNs. Last but not the least, with an emerging trend of "we-media" (aka "self-media"), long-tail users are eager to know which topics to be published would receive more attentions from their potential SN audiences.

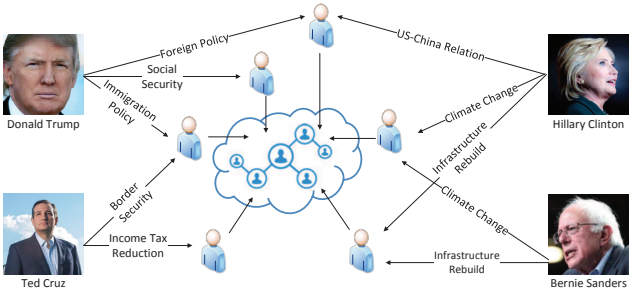


Figure 1: Campaign Propagation Network.

Motivated by the new demand, we propose a novel social influence query known as *personalized social influential tag exploration* (PITEX): Given a target user, it extracts a size-k tag set that maximizes the user's social influence from a set of tags which characterize the content propagated in an SN. We formalize the problem of answering a PITEX query by adopting the existing topic-based social influence model [2, 16, 26, 6]: we generate a total set of tags from the contents propagated in an SN, and derive the correlation between the tags and influence spread from the propagation history among SN users. Based on this, we introduce a probabilistic model to compute social influence of each tag set w.r.t. the

SIGMOD'17, May 14-19, 2017, Raleigh, NC, USA

DOI: http://dx.doi.org/10.1145/3035918.3035952

target user. Moreover, we analyze the complexity of finding the best tag set that maximizes a user's influence: we prove that it is not only a NP-hard problem but also NP-hard to be approximated within any constant ratio.

In order to support novel *online* applications, such as instantly suggesting influential tags once any user on Twitter wishes to post viral ads, it calls for high or even real-time performance to answer PITEX queries. To address this challenge, we introduce a sampling-based framework to estimate the influence spread of each size-k tag set by sampling edges in the social graph, and then find the one with maximum influence. We first identify that Monte Carlo (MC) sampling [19, 21] and Reverse Random set (RR) sampling [5, 36], which are state-of-the-art sampling techniques in the existing studies for influence maximization, have serious performance drawbacks when employed in our framework as they may sample many unnecessary edges. Then, we devise a *lazy propagation sampling* algorithm to sample as few edges as possible, which significantly outperforms MC and RR. We also develop a best-effort exploration strategy that effectively prunes tag sets associated with different influence graphs. To enable real-time influence computation, we propose an effective index structure, which materializes the "influencer" of uniformly sampled users to avoid the expensive process of online sampling. Furthermore, a novel filter and verification framework is proposed on the index structure to efficiently identify users who are effectively influenced through a tag set. We also develop a delay materialization technique to support fast influence computation with moderate index size. We prove that all the approaches proposed achieve a $\frac{1-\varepsilon}{1+\varepsilon}$ approximate solution, and the index-based methods are faster than the online sampling methods by orders of magnitudes.

To summarize, we make the following contributions.

(1) To the best of our knowledge, we are the first to study personalized social influential tags exploration. We formalize the problem and prove its hardness (Sec. 3).

(2) We introduce a sampling-based framework and employ MC and RR sampling to answer PITEX. We theoretically analyze these two strategies and pinpoint the drawbacks of applying both strategies in processing PITEX. (Sec. 4).

(3) We devise lazy propagation sampling and best-effort exploration for optimizing online sampling (Sec. 5). To enable real-time processing, we develop an index structure together with effective pruning and materialization techniques. The proposed approaches can significantly reduce the running time while preserving the theoretical bound (Sec. 6).

(4) We evaluate the performance on four large-scale real datasets. Results of our experimental study have shown the efficiency and effectiveness of our methods (Sec. 7).

2. RELATED WORK

Topic-Based Propagation Models. Topic-aware propagation models [34, 26, 2] were proposed to capture the fact that users' influence are topic-dependent. Tang et al. [34] studied a problem of learning user-to-user topic-wise influence strength. Subsequently, probabilistic models, e.g., [26], for joint inference of the topic distribution and topic-wise influence strength were proposed. Recently, Barbieri et al. [2] improved previous works by extending the classic IC model to a Topic-aware IC (TIC) model. PITEX employs these models to not only formalize the propagation process, but also extract tags as well as their correlation to social influence. On the flip side, PITEX is different as it focuses on tags, which are textual keywords that can be easily interpreted by end users, instead of topics derived from topic-modeling mechanisms represented as numerical variables.

Influence Maximization (IM) and Topic-aware IM. IM is an important and well-studied problem in the areas of social influences [6, 19, 36]. It finds a seed set of k most influential users on a propagation network. Recently, a topic-aware IM problem was proposed in [2] to find influential seeds by considering a topic-based propagation model. Subsequently, many algorithmic solutions [6, 7, 28] were devised to improve the efficiency. PITEX is fundamentally different from topic-aware IM, as we focus on selecting tags that maximize a user's social influence, instead of k most influential users. This essentially results in different characteristics of the underlying graphs. In topic-aware IM, the graph is *fixed* in terms of activation probabilities given a query. On the contrary, in PITEX, each candidate tag set corresponds to *a distinct graph*, as activation probabilities are different under different tag sets. Nevertheless, we can adapt influence estimation techniques in the existing work to solve PITEX, such as sampling-based methods, e.g., MC [19] and RR [5], and tree-based methods [7, 6]. In this paper, we analyze limitations of these methods theoretically and empirically, and develop novel techniques, such as lazy propagation sampling and index-based strategies, which are far more efficient.

Query processing In Uncertainty Graphs. Query processing over uncertainty graphs has attracted much attention [20, 18, 29, 30]. To model an uncertainty graph, a *possible world* semantic is often used, where each edge is associated with a probability that indicates likelihood of its existences [20, 18, 29, 30]. Among all queries proposed over uncertainty graphs, reliability query is most similar to PITEX proposed in this paper. Jin et al. [18] proposed a distance-constraint reachability problem, which computes the probability that distance from two input vertices is less than or equal to a user-defined threshold. Khan et al. studied a reliability set query [20] that finds all vertices to be reached from a source vertex with a probability higher than a user defined threshold. The reliability queries are different from PITEX as their input graph is fixed whereas the propagation graph in PITEX changes dramatically when different tag set is used. Thus the indices and optimization methods used in the existing work cannot be adopted.

Social Recommender Systems. Social recommendation [31, 32, 12, 1, 22, 33, 27, 10, 24] is a popular area in recommender systems research. Existing approaches either recommend contents that match users' interests [31, 32, 12], or recommend prevalent contents with high social popularity [1, 22, 33]. Although these approaches also output tags, they cannot capture the features supported by PITEX, since both contents that match a user's interests and prevalent contents could not match the influential characteristics.

3. THE PITEX PROBLEM

3.1 Problem Definition

Topic-Aware Influence Model. We model an SN as a directed graph $G(V, E)$, where V is a set of users and each edge $e = (u, v) \in E$ captures the friendship or follow relationship from u to v. To model how information is propagated in an SN, we adopt the extensively studied topic-aware propagation model [2, 16, 26, 6]. Intuitively, the model extracts a

set of hidden topics $Z = \{z_1, z_2, \ldots, z_{|Z|}\}$ from social activities (tweets and replies) propagated on a SN, and considers social influence depends on the topics. Formally, given an edge $e = (u, v)$, a *topic-aware influence probability* $p(e|z)$ is introduced to model how u *influences* v under topic $z \in Z$. This type of probabilities can be learnt from a "log of past propagation" in the SN [2].

Influence of Tag Sets. Despite of the success in IM [6, 25, 28], topics are not sufficient in helping users to explore their "selling-points", i.e., how they influence the SN. The reason is that the topics are essentially probabilistic distributions in a *latent numerical space*, which cannot be easily interpreted if presented to users for their exploration. Thus, we naturally introduce *tags* to provide more comprehensive exploration. Tags are commonly used in many social networks that allow users to characterize their content [31, 32, 12], such as *hashtags* in Twitter and Instagram. Even if users do not explicitly provide tags, one can extract representative (e.g., most frequent) *keywords* from the content to produce tags. Take Fig. 1 as an example: we can extract tags of `Hillary Clinton`'s tweets by collecting the occurring hashtags or analyzing the keywords, which are easily interpreted and useful to direct the exploration process.

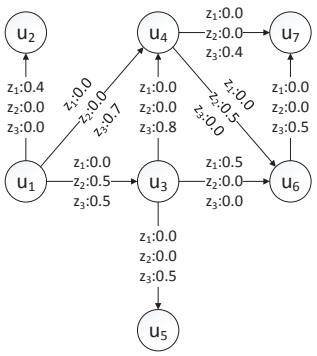
Formally, we use Ω to denote a set of tags in an SN. Following the convention of topic modeling studies [2, 4], we consider that tags are distributed over topics by introducing probability $p(w|z)$ that represents the likelihood of sampling tag w from topic z. Note that we can employ the learning algorithms in [2] to jointly learn both probabilities $p(w|z)$ and $p(e|z)$ from a log of past propagation. Next, consider any propagated content (e.g., a tweet) described by a set $W \in \Omega$ of tags. We compute the influence probability of edge e, denoted by $P(e|W)$, by combining probabilities $p(w|z)$ and $p(e|z)$ according to the bag-of-words bayesian language model [2, 16, 4, 12], i.e.,

$$p(e|W) = \sum_{z \in Z} p(e|z) \cdot p(z|W) = \sum_{z \in Z} p(e|z) \cdot \frac{p(W|z)p(z)}{p(W)}$$
$$= \sum_{z \in Z} p(e|z) \cdot \frac{\prod_{w \in W} p(w|z)p(z)}{\sum_{z' \in Z} \prod_{w \in W} p(w|z')p(z')}, \qquad (1)$$

where $p(z)$ denotes the prior probability.

Based on $p(e|W)$ in each edge, we present the influence in an SN by considering the propagation process. In this paper, we use the independent cascade (IC) model [7, 19] as it has been widely adopted[1]. In the IC model, each user in V is either in an *active* state or *inactive* state. Initially every user is inactive. Then, given a tag set $W \subseteq \Omega$, a *target* user u is selected to become active at time step 0. Next, each active user at time step i attempts to activate the inactive neighbors with $p(e|W)$, i.e., the likelihood of v being activated by u is $p(e|W)$. At time step $i+1$, the newly activated users from step i will try to activate their inactive neighbors. This process terminates when no more users are activated. Finally, we introduce $\mathcal{I}(u|W)$, which is the number of active users after the process terminates, to evaluate the *influence spread* of tag set W w.r.t. user u. Before proposing the problem, we list all the frequently used notations in Table 1 to facilitate the presentation.

[1]The approaches proposed in this paper can also support other propagation models, such as linear threshold model [14] and the more general triggering model [19, 36]



$p(w_i\|z_j)$	z_1	z_2	z_3
w_1	0.6	0.4	0.0
w_2	0.4	0.6	0.0
w_3	0.0	0.4	0.6
w_4	0.0	0.4	0.6

$p(z_j\|\{w_a, w_b\})$	z_1	z_2	z_3
$\{w_1, w_2\}$	0.5	0.5	0.0
$\{w_1, w_3\}$	0.0	1.0	0.0
$\{w_1, w_4\}$	0.0	1.0	0.0
$\{w_2, w_3\}$	0.0	1.0	0.0
$\{w_2, w_4\}$	0.0	1.0	0.0
$\{w_3, w_4\}$	0.0	0.33	0.67

(a) Topic-aware social graph (b) Tag-topic probabilities

Figure 2: A running example of PITEX

Table 1: Frequently used notations in the paper

$G(V, E)$	the social graph with vertex and edge sets
Ω, Z	the global tag set and topic set
$W \subseteq \Omega$	candidate tag set
k	number of tags selected for a PITEX query
$\mathbb{E}[\mathcal{I}(u\|W)]$	expected influence spread of user u given $p(e\|W)$
$\mathcal{R}_W(u)$	set of vertices reached by u on G by removing each edge s.t. $p(e\|W) = 0$
$E_W(u)$	$\forall e = (u, v) \in E_W(u)$ satisfies $u, v \in \mathcal{R}_W(u)$

Personalized Social Influential Tags Exploration. Our problem is formally defined as follows.

Definition 1. *(Personalized Social Influential Tags Exploration (*PITEX*)) Given a social network G, a* PITEX *query consists of a target user u who is initially activated and a number k, and it aims to find a k-size tag set W^* that maximizes u's expected influence spread, i.e., $W^* = \arg_{W \subseteq \Omega, |W|=k} \max \mathbb{E}[\mathcal{I}(u|W)]$.*

Example 1. *Figure 2 shows an example, consisting of a social graph with topic-based influence probabilities on each edge, and a tag topic probability $p(w|z)$ table. Under a uniform prior $p(z_j) = \frac{1}{|Z|}$, influence probability of edge $e = (u_1, u_2)$ w.r.t. tag set $\{w_1, w_2\}$ is computed as $p(e|\{w_1, w_2\}) = 0.4 \cdot 0.5 + 0 \cdot 0.5 + 0 \cdot 0 = 0.2$. The influence spread of u_1 across the entire social graph w.r.t $\{w_1, w_2\}$ is $\mathbb{E}[\mathcal{I}(u_1|\{w_1, w_2\})] = 1.5125$. When u_1 issues a* PITEX *query for two tags ($k = 2$), $W^* = \{w_3, w_4\}$ will be selected as the result since it generates the maximum expected influence spread $\mathbb{E}[\mathcal{I}(u_1|W)]$ from u_1 among all the size-2 tag sets.*

3.2 Problem Hardness

This section analyzes the complexity of PITEX. It turns out that not only this problem is NP-hard, but also there exists *no constant approximation algorithm* unless NP=P. To prove this claim, we first show the following lemma.

Lemma 1. *(k-label s-t reachability) Consider a directed multi-graph $G = (V, E, \mathcal{L})$ where V is a set of vertices, E is a set of edges, and $\mathcal{L}$ is a set of labels. Each edge $e \in E$ is associated with a label $l \in \mathcal{L}$. Given a pair of vertices s, t in G and a number k, the problem, which checks if there exists a label set $L \subseteq \mathcal{L}$ and $|L| = k$ s.t. s reaches t in the subgraph of G induced by L is NP-hard.*

The proof can be found in Appx. B.1. With Lemma 1, we present Theorem 1 to show the hardness of PITEX.

Theorem 1. *Given an instance $\langle G, u, k \rangle$ of* PITEX*, it is NP-hard to obtain a solution that achieves an expected*

influence spread of at least $\frac{1}{\sqrt{n}}\texttt{OPT}$ *where* n *is the number of vertices in* G *(*$n = |V|$*) and* $\texttt{OPT}$ *is the maximum influence spread of* u *under any* k*-size tag set.*

PROOF. We prove by a reduction from the k-label s-t reachability problem. Given an instance $\langle G = (V, E, \mathcal{L}), k, s \in V, t \in V\rangle$ of the k-label s-t reachability problem, we construct an influence graph G' as follows. First, we construct the vertex set of G' as $V \cup V'$ ($V \cap V' = \emptyset$) where $|V'| = n^2 - n$ and $V' = \{u'_1, u'_2, \ldots, u'_{n^2-n}\}$. Second, we construct a tag set $\Omega = \{w\}$ and a topic set $Z = \{z\}$ in G' such that $|\Omega| = |Z| = |\mathcal{L}|$, $p(w_i|z_i) = 1$ for $i = 1, \ldots, |\Omega|$, and $p(w_i|z_j) = 0$ for any $i \neq j$. Third, for each edge $e = (u_i, u_j) \in E$ associated with a label l_t in G, we add an edge to the corresponding vertices u_i and u_j in G' and set $p(e|z_t) = 1$. In addition, we also create another set E' of edges for G' as $E' = \{(t, u'_1), (u'_1, u'_2), \ldots, u'_{n^2-n-1}, u'_{n^2-n}\}$, and for each $e \in E'$, we set influence probabilities $p(e|z) = 1$, $\forall z \in Z$. Fourth, we take s as query user in G'.

With the above polynomial time reduction, we prove the theorem by contradiction. Given the constructed graph G' with n^2 vertices, assume that there exists a polynomial time algorithm $\mathcal{A}$ that can solve PITEX and achieves an influence spread of at least $\texttt{OPT}' = \texttt{OPT}/\sqrt{n^2}$. Then, we can create another algorithm to solve the k-label s-t reachability problem $\langle G = (V, E, \mathcal{L}), k, s, t\rangle$ by simply considering two cases.

Case 1: If the produced influence spread $\texttt{OPT}' \leq n - 1$, we must have $\texttt{OPT} \leq n - 1$, which can be proved by contradiction: If $\texttt{OPT} > n - 1$, it means vertex s must reach vertex t. According to the construction of E', s must also influence all the vertices in V', and thus we have $\texttt{OPT} \geq n^2 - n + 2$ (i.e., $n^2 - n$ vertices in V' and s, t). As $\texttt{OPT}/\sqrt{n^2} \leq \texttt{OPT}' \leq n - 1$, we have $n^2 - n + 2 \leq \texttt{OPT} \leq n \cdot (n - 1)$, which induces an incorrect result $2 \leq 0$. Now that we know $\texttt{OPT} \leq n - 1$, it is trivial to see that s cannot reach t in graph G.

Case 2: If the influence spread $\texttt{OPT}' > n - 1$, we also have $\texttt{OPT} \geq \texttt{OPT}' > n - 1$. Based on the above analysis, we must have $\texttt{OPT} \geq n^2 - n + 2$. As $n^2 - n + 2 > n - 1$ always holds, we can see s can reach t in G.

However, the k-label s-t reachability problem is NP-hard (Lemma 1). Thus, if there exists an algorithm that solves PITEX and achieves expected influence spread of at least $\texttt{OPT}/\sqrt{|V|}$, then P=NP. □

4. SAMPLING-BASED FRAMEWORK

One straightforward solution to PITEX is to enumerate all possible size-k tag sets and select the one with the maximum influence spread. However, the computation of influence spread $\mathbb{E}[\mathcal{I}(u|W)]$ of user u for any tag set W is expensive due to its #P-hardness [7]. We introduce a *sampling-based framework* on top of the enumeration-based approach to achieve a tight approximation ratio. Before presenting the details, readers are referred to Appx. B.2 for the Chernoff bounds [8] which are frequently used in our analysis.

Given a social graph G, a user u and a number k, the framework enumerates every possible k-size tag set $W \subseteq \Omega$, and computes the influence spread $\mathbb{E}[\mathcal{I}(u|W)]$ of u under tag set W. Specifically, it employs sampling based strategies to implement EstimateInfluence (see Algo. 1). It first computes the size of samples θ_W (SampleSize) and then estimates influence spread $\widehat{\mathbb{E}}[\mathcal{I}(u|W)]$ (SampleEstimate) based on the samples. Finally, the framework returns the tag set W^* with the maximum estimated influence.

Algorithm 1: EstimateInfluence $(u, G, W, \varepsilon, \delta)$

1 $\theta_W \leftarrow$ SampleSize (ε, δ) ;

2 **return** $\widehat{\mathbb{E}}[\mathcal{I}(u|W)] \leftarrow$ MCSample/RRSample (u, G, θ_W) ;

To support the framework, we can adopt existing sampling strategies. Next, we present two state-of-the-art strategies and then discuss how to determine the sample size θ_W.

Monte Carlo Sampling. Inspired by [19], we can use a Monte Carlo (MC) method for computing $\widehat{\mathbb{E}}[\mathcal{I}(u|W)]$. It generates an instance of sample g from G as follows. It starts from u and traverses G by removing each edge with a probability $1 - p(e|W)$ until no vertex can be reached. Let $\mathcal{I}_g(u|W)$ denote the number of nodes reachable from u in g. Suppose we generate multiple sample instances, i.e., $\{g_1, g_2, \ldots, g_{\theta_W}\}$. The expected influence spread is then estimated by: $\widehat{\mathbb{E}}_{\text{MC}}[\mathcal{I}(u|W)] = \sum_{i=1}^{\theta_W} \mathcal{I}_{g_i}(u|W)/\theta_W$.

Reverse Reachable Set Sampling. We can also apply the recently proposed reverse reachable set (RR) sampling [5] to estimate $\widehat{\mathbb{E}}[\mathcal{I}(u|W)]$. It first uniformly samples a vertex v_i from the set $\mathcal{R}_W(u)$ of vertices that can be reached by u on G by removing each edge if $p(e|W) = 0$, and then generates a subgraph g_i from G by removing each edge e with a probability $1 - p(e|W)$. If u reaches v_i on g_i, we set $\mathbf{1}[u \rightsquigarrow v_i] = 1$; otherwise, $\mathbf{1}[u \rightsquigarrow v_i] = 0$. By sampling multiple vertices $v_1, \ldots, v_{\theta_W}$ from $\mathcal{R}_W(u)$, the influence spread is estimated by: $\widehat{\mathbb{E}}_{\text{RR}}[\mathcal{I}(u|W)] = \sum_{i=1}^{\theta_W} \mathbf{1}[u \rightsquigarrow v_i] \cdot |\mathcal{R}_W(u)|/\theta_W$. Complementary examples for MC and RR can be found in the extended version of this paper [23].

A critical issue in the sampling strategies is to determine sufficient number θ_W of sample instances, which works for any tag set W to ensure estimation accuracy. Existing works on RR sampling [36] have provided the following bound.

LEMMA 2. *[36] Whenever* θ_W *satisfies*

$$\theta_W = \frac{2 + \varepsilon}{\varepsilon^2} \cdot |\mathcal{R}_W(u)| \cdot \frac{\log(\delta) + \log\binom{|\Omega|}{k} + \log 2}{\mathbb{E}[\mathcal{I}(u|W)]}, \qquad (2)$$

then $|\widehat{\mathbb{E}}_{\text{RR}}[\mathcal{I}(u|W)] - \mathbb{E}[\mathcal{I}(u|W)]| < \varepsilon \cdot \mathbb{E}[\mathcal{I}(u|W)]$ *holds with a probability of at least* $1 - \delta^{-1}\binom{\Omega}{k}^{-1}$ *for the* RR *method.*

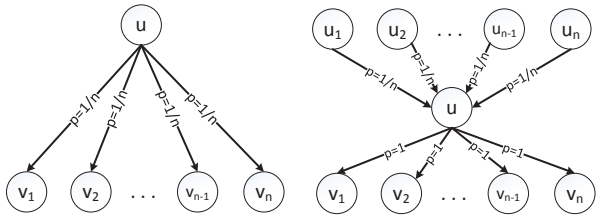
According to our studies, the above sample size can also be applied to MC when solving PITEX, i.e.,

LEMMA 3. *Whenever* θ_W *satisfies Eqn. (2), the inequality:* $|\widehat{\mathbb{E}}_{\text{MC}}[\mathcal{I}(u|W)] - \mathbb{E}[\mathcal{I}(u|W)]| < \varepsilon \cdot \mathbb{E}[\mathcal{I}(u|W)]$ *holds with a probability of at least* $1 - \delta^{-1}\binom{\Omega}{k}^{-1}$ *for the* MC *method.*

The proof can be found in Appx. B.3. In summary, Lemmas 2 and 3 show that both MC and RR sampling methods may use the same sample size θ_W for the solutions to achieve the same error bound. We show the approximation ratio of the proposed enumeration-based approach, as formally stated in the following (Proof in Appx. B.5).

THEOREM 2. *The enumeration-based method always has a* $\frac{1-\varepsilon}{1+\varepsilon}$ *approximation ratio to the optimal solution to the* PITEX *problem with a probability of* $1 - \delta^{-1}$.

Theoretical Analysis. Interestingly, we find out that the framework using either MC or RR sampling to solve PITEX could lead to inefficient solutions due to the unexpected large complexities to generate *one sample instance*. As we have



(a) MC (b) RR

Figure 3: Examples where MC and RR have high computational complexities.

shown that both methods use the same θ_W to achieve the same error bound, to prove the above claim, clearly we only need to analyze the expected time complexity of computation for *one sample instance* of MC and RR.

The complexity can be naturally measured by the expected number of edges visited by MC (RR) sampling for one sample instance, denoted by $\texttt{ENE}^{\text{MC}}$ ($\texttt{ENE}^{\text{RR}}$). Existing work [36] has shown the following lemma for $\texttt{ENE}^{\text{RR}}$.

LEMMA 4. *[36] Given the vertex set $\mathcal{R}_W(u)$ reachable by u by removing each edge with $p(e|W) = 0$ from G and the set $(E_W(u))$ of edges connecting vertices in $\mathcal{R}_W(u)$, we have*

$$\texttt{ENE}^{\text{RR}} = O(|E_W(u)| \cdot \mathbb{E}[\mathcal{I}(v^{in} \rightsquigarrow v^*|W)]) \quad (3)$$

where $\mathbb{E}[\mathcal{I}(v^{in} \rightsquigarrow v^*|W)])$ is the influence probability from a randomly selected vertex v^{in}, with the probability of selecting v^{in} proportional to its in-degree, to activate another uniformly selected vertex v^*.

As no prior work has shown any result for $\texttt{ENE}^{\text{MC}}$ of MC sampling, we present the following lemma (Proof in Appx. B.4).

LEMMA 5. *Let $\mathbb{E}[\mathcal{I}(u \rightsquigarrow v^{ot}|W)]$ denote the influence probability from the query user u to activate a randomly selected vertex v^{ot} with the probability of selecting v^{ot} proportional to its out-degree.*

$$\texttt{ENE}^{\text{MC}} = O\left(|E_W(u)| \cdot \mathbb{E}[\mathcal{I}(u \rightsquigarrow v^{ot}|W)]\right) \quad (4)$$

Now, we are ready to analyze the time complexity of the sampling strategies by multiplying $\texttt{ENE}^{\text{MC}}$ ($\texttt{ENE}^{\text{RR}}$) with θ_W analyzed in lemmas 2-5. Let $\Lambda = \frac{2+\varepsilon}{\varepsilon^2}(log(\delta) + \log\binom{|\Omega|}{|k|} + log2)$ and $\mathbb{E}[\mathcal{I}(u \rightsquigarrow v^*|W)]$ be the influence probability from u to activate an uniformly selected vertex $v^* \in \mathcal{R}_W(u)$. we can see that the complexity of the enumeration method with MC is $O\left(\Lambda \cdot |E_W(u)| \cdot \frac{\mathbb{E}[\mathcal{I}(u \rightsquigarrow v^{ot}|W)]}{\mathbb{E}[\mathcal{I}(u \rightsquigarrow v^*|W)]}\right)$. On the other hand, the complexity of RR is $O\left(\Lambda \cdot |E_W(u)| \cdot \frac{\mathbb{E}[\mathcal{I}(v^{in} \rightsquigarrow v^*|W)]}{\mathbb{E}[\mathcal{I}(u \rightsquigarrow v^*|W)]}\right)$.

Note that in the scenario where $\frac{\mathbb{E}[\mathcal{I}(u \rightsquigarrow v^{ot}|W)]}{\mathbb{E}[\mathcal{I}(u \rightsquigarrow v^*|W)]}$ is a constant w.r.t. G, MC produces a linear time algorithm. The similar conclusion holds for RR when $\frac{\mathbb{E}[\mathcal{I}(v^{in} \rightsquigarrow v^*|W)]}{\mathbb{E}[\mathcal{I}(u \rightsquigarrow v^*|W)]}$ is also a constant. However, the following two realistic counterexamples pinpoint scenarios where both methods may have quadratic complexities for even a sparse graph, which are prohibitive for processing real-world large SNs.

EXAMPLE 2. *In Fig. 3 (a), an input graph consists of a root vertex u which has an influence edge to each of the remaining n vertices with a probability of $\frac{1}{n}$. This example pictures the situation where a user who has a lot of followers but has a low impact in the social network.*

When u is the query vertex, we have $\mathbb{E}[\mathcal{I}(u \rightsquigarrow v^{ot}|W) = 1 \cdot 1 + \sum_{i \in 1..n} 0 \cdot \frac{1}{n} = 1$ and $\mathbb{E}[\mathcal{I}(u \rightsquigarrow v^*|W)] = \frac{1}{n+1} \cdot 1 + \sum_{i \in 1..n} \frac{1}{n+1} \cdot \frac{1}{n} = \frac{2}{n+1}$. Under such scenario, $\frac{\mathbb{E}[\mathcal{I}(u \rightsquigarrow v^{ot}|W)]}{\mathbb{E}[\mathcal{I}(u \rightsquigarrow v^*|W)]} = O(n)$, which brings quadratic complexity to MC.

EXAMPLE 3. *In Fig. 3 (b), an input graph consists of a central vertex v which has an influence edge to n vertices with a probability 1. There is another set of n vertices which has an influence probability of $\frac{1}{n}$ to v. The example presents the situation where a celebrity who has a huge number of followers and also follows a lot of users in the social network. However, the influence from a celebrity to a normal individual is trivially much higher than the other way around.*

When any u_j is the query vertex, we have $\mathbb{E}[\mathcal{I}(v^{in} \rightsquigarrow v^*|W) = \frac{n}{2n} \cdot (\frac{1}{2n+1} \cdot 1 + \sum_{i=1..n} \frac{1}{2n+1} \cdot 1) + \sum_{i=1..n} \frac{1}{2n} \cdot 1 = \frac{3n+2}{4n+2}$ and $\mathbb{E}[\mathcal{I}(u_j \rightsquigarrow v^*|W)] = \frac{1}{2n} \cdot 1 + \frac{1}{2n}\frac{1}{n} + \sum_{i=1..n} \frac{1}{2n}\frac{1}{n} = \frac{1}{n} + \frac{1}{2n^2}$ for any u_j. Under such scenario, $\frac{\mathbb{E}[\mathcal{I}(v^{in} \rightsquigarrow v^*|W)]}{\mathbb{E}[\mathcal{I}(u_j \rightsquigarrow v^*|W)]} = O(n)$, which brings quadratic complexity to RR.

Given the above counter examples, MC and RR could both take $O(\Lambda \cdot |E_W(u)| \cdot |\mathcal{R}_W(u)|)$ to evaluate the influence score which is obviously not scalable as $|\mathcal{R}_W(u)| = O(|V|)$ and $|E_W(u)| = O(|E|)$ in highly connected social graphs. In the following section, we develop a series of optimized sampling methods to speed up the influence estimation process.

5. OPTIMIZING ONLINE SAMPLING

To reduce the sampling complexity in the above framework, this section develops two techniques, *lazy propagation sampling* and *best effort exploration*. The former reduces the sampling cost when evaluating influence spread $\mathbb{E}[\mathcal{I}(u|W)]$ for any tag set W, while the latter improves the enumeration-based approach by pruning a large number of tag sets without evaluating their influence spread.

5.1 Lazy Propagation Sampling

According to our previous analysis, the main reason why MC and RR fail to deliver efficient influence estimation is the unexpected high complexity to generate one sample instance which has to probe a large number of edges. For example, in Fig. 3 (a), when MC is applied to estimate u's influence, the generation of *each individual* sample instance has to probe *every* u's *out-going* edge e to test if e is activated (with a probability $p(e|W)$). Obviously, this process will lead to many "unnecessary" probings for the unactivated edges, as existing works on learning propagation probabilities in real-world SNs often deliver sparse influence graphs [2, 13], i.e., the propagation probability is low for a large number of edges. Similarly, in Fig. 3 (b), each sample instance generated by RR has to probe all v's *in-coming* edges, even though the activation probabilities are low. Thus the key to speed up the influence estimation is to avoid probing as many such "unnecessary" edges as possible. Towards this goal, we develop a *lazy propagation sampling* strategy that only probes the edges when they are activated.

The key challenge is to predict the case in which an edge e will be activated. According to the sampling process, the events of any edge e being activated across θ_W sample instances can be considered as independent uniform random variables (r.v.s), each of which is equivalent to a coin toss with a head probability of $p(e|W)$. This means the edge will become activated (and thus needs to be probed) once a head appears after a number of tosses, which is literally a *geometric distribution* with parameter $p(e|W)$. Based on

Algorithm 2: LazySample $(u, G, \theta_W, \delta, \epsilon)$

```
1  s ← 0;
2  for i = 1, ..., θ_W do
3      h ← {u}; // The traversal frontier.
4      while h ≠ ∅ do
5          v ← h.pop();
6          if v is not initialized then
7              c_v ← 0, h_v ← ∅
8              for nbr ∈ v's neighbour sets do
9                  X(nbr) ← geometric r.v. instance
10                 h_v ← h_v ∪ {⟨, nbr, X(nbr)⟩}
11         while h_v.top() == c_v do
12             ⟨nbr, X(nbr)⟩ ← h_v.pop();
13             h ← h ∪ {nbr} if nbr is not visited.
14             X'(nbr) ← geometric r.v. instance
15             h_v ← h_v ∪ {⟨nbr, X'(nbr) + c_v⟩}
16         c_v ← c_v + 1; s ← s + 1;
17     if s/|R_W(u)| ≥ 1 + (1+ε)√2/ε² · log(2/(δ·(Ω choose k))) then
18         break
19     Reset all visited vertex to not visited
20 return s/θ_W
```

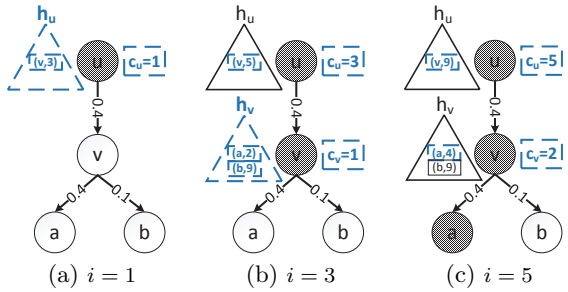
this observation, to decide when to probe edge e, we only need to sample a geometric r.v. that determines the next sampling instance in which e is activated. For example, suppose that θ_W of the MC sample instances are generated by starting from u in Fig. 3 (a). Instead of probing every edge e from u to v_i for θ_W instances, our proposed strategy first samples a geometric r.v., say 3, for edge e, which means e will not be activated until the third instance. Thus, it can safely skip probing e in the first two instances. Then, after the third instance, we sample a geometric r.v. again to determine the next sample instance that probes e. In this way, our strategy only probes the edge $\frac{\theta_W}{n}$ instead of θ_W times in expectation. The method is also applicable for RR and one can picture that the reverse probings from v to each of its in-coming neighbors in Fig. 3 (b) is reduced by n times.

To show the above strategy is correct, we establish the equivalence between the number of heads observed by θ_W tosses (the number of times an edge is activated in our problem) and the number of successes indicated by a sequence of geometric r.v.s with their sum smaller or equal to θ_W.

LEMMA 6. *The following two r.v.s are statistically identical: 1) the number of heads observed by θ Bernoulli trials with success probability of p; 2) a random number Y s.t. $\sum_{i=1..Y} X_i \leq \theta$ and $\sum_{i=1..Y+1} X_i > \theta$ where X_is are i.i.d geometric r.v.s with parameter p.*

Lemma 6 is proved in Appx. B.4 B.6, which indicates there is no statistical difference between using the lazy sampling method and MC/RR.

The pseudo-code of the optimized MC with lazy propagation sampling is presented in Algo. 2. Above all sampling instances, for any $v \in \mathcal{R}_W(u)$, it initializes a heap h_v and pushes every v's out-going neighbor nbr with a geometric r.v., i.e., $\langle nbr, X(nbr)\rangle$ into h_v (lines 6-10). It also maintains a counter c_v to keep track of the number of time v has been visited. Once an r.v. in h_v is equal to c_v, the corresponding neighbor will be probed, and then a new r.v. is generated to decide the next time to probe the neighbor (See Lines 11-15). Lastly, we impose a stopping condition to terminate the sampling process early(See Line 17). The intuition is that, when the sum of observed random values



(a) $i = 1$ (b) $i = 3$ (c) $i = 5$

Figure 4: Example of lazy propagation algorithm.

are sufficiently large, the estimation can be stopped early since the true value has a large error tolerance range. The proof that such stopping condition ensures the same approximation guarantee as Theorem 2 can be easily established by existing works on martingale simulations [35].

EXAMPLE 4. *Fig. 4 shows a simplified example for running the lazy propagation on an influence graph. Let us assume the sampling process always starts from u. The shaded nodes are the ones visited during a particular sampling iteration and the updated values are marked in red. In iteration $i = 1$, u is initialized with h_u containing its neighbor v and a geometric instance, i.e. 3. This means the next visit to v is in the third visit, and thus this iteration can safely terminate. It is only in iteration $i = 3$ where c_u is updated to 3 as u is visited in each of the last three sampling processes, that u visits v and re-assigns v with another geometric r.v. Meanwhile, v is visited for the first time (c_v is set to 1) so it will be initialized with h_v containing its neighbors a and b. As the earliest visit to v's neighbor happens in the second visit of v, the sampling iteration terminates at this point. Finally, a will be visited in the iteration $i = 5$.*

LEMMA 7. *Let $\texttt{ENE}^{\text{LP}}$ denote the expected number of edges visited by a lazy propagation instance under the IC model, the following holds:*

$$\texttt{ENE}^{\text{LP}} = O(|\mathcal{R}_W(u)| \cdot \mathbb{E}[\mathcal{I}(u \rightsquigarrow v^*|W)])$$

With Lemma 7, one can easily see the complexity for the lazy propagation method is $O\left(\Lambda \cdot |\mathcal{R}_W(u)|\right)$ where $\Lambda = \frac{2+\varepsilon}{\varepsilon^2}(log(\delta) + \log\binom{|\Omega|}{|k|}) + log2)$, which shows that lazy propagation sampling is indeed more efficient than MC and RR as it visits much smaller number of edges.

5.2 Best-Effort Exploration

Although enumeration-based sampling solves PITEX with high accuracy, it has to evaluate all size-k tag sets, which is exponential against k. Moreover, the complexity of sampling strategies for each tag set W is also expensive. This motivates us to develop effective pruning methods in order to avoid evaluating all k-size tag sets.

To this end, we introduce a *best-effort* exploration strategy. The basic idea of this strategy is to progressively select a tag w into a partial solution tag set, i.e., W with $|W| < k$. Then, it estimates the *upper bound* of the influence spread for any k-size tag set containing the partial solution W. Obviously, if the newly selected tag makes the partial solution set impossible to be optimal (i.e., the upper bound is already smaller than a known solution), we can prune all size-k tag sets containing this partial solution. The essential challenge for the best effort exploration is to estimate the upper bound

for the influence w.r.t. any partial tag set. In this paper, we introduce a bound estimation by considering two scenarios for the tag-topic distribution ($p(w|z)$): 1) $p(w|z)$ is sparse, meaning a large number of tags have no probability to appear in many topics; 2) $p(w|z)$ is dense, meaning the topics are often expressed in most of the tags. Under the aforementioned scenarios, we have the following lemma of the upper bound of the influence spread for a partial tag set W.

LEMMA 8. *Given a partial tag set W with $|W| < k$, an upper bound of the influence probability for each edge $e \in E$, denoted by $p^+(e|W)$, can be estimated as:*

$$p^+(e|W) = \min\Big(\max_{\substack{p(z|W)>0 \\ z\in Z}} p(e|z), \tag{5}$$

$$\sum_{\substack{p(z|W)>0 \\ z\in Z}} p(e|z) \cdot \max_{\substack{W^*\in\Omega\setminus W \\ |W^*|+|W|=k}} \prod_{w\in W\cup W^*} \frac{p(w|z)p(z)}{\prod_{z'\in Z} p(w|z')^{p(z')}}\Big) \tag{6}$$

W.L.O.G, $p^+(e|\emptyset) = \max_{z\in Z} p(e|z)$.

We will show in Appx. B.8 that given a tag set W, $p^+(e|W) \geq p(e|W')$ for any $W' \supset W$ s.t. $|W'| = k$. Examples for Lemma 8 can be found in [23]. Based on the bounds $p^+(e|W)$ of edge probability, we can devise our best exploration strategy to enable early termination for PITEX. Interested readers could refer the Appx. C for the detailed algorithm and the theoretical analysis.

6. INDEX-BASED INFLUENCE ESTIMATION

Although the online optimization strategies in the previous section can significantly improve the performance, they still cannot enable *online* exploration, as influence estimation incurs high computational complexities, and, even worse, the estimation may be invoked many times. To address this issue, this section presents more efficient influence estimation. We first introduce an index-based estimation method in Sec. 6.1, and then discuss pruning strategies and an efficient materialization technique for reducing the index size in Sections 6.2 and 6.3 respectively.

6.1 RR-Graph Index Structure

The limitation of our previous influence estimation is that it has to regenerate samples on-the-fly for each user and tag set, which is very expensive. This motivates us to develop a more efficient *index-based* method. Different from the previous one, this method performs sampling in an offline manner and constructs an index structure over the obtained samples. For online estimation, given any user and any tag set, it estimates the influence spread based on the constructed index, instead of regenerating the samples.

The design of the index structure is inspired by the RR sampling. Recall that RR sampling uniformly samples a vertex v_i and checks if u influences v_i w.r.t. tag set W. As many vertices sampled may be far away from u, redundant samples are generated. However, such a disadvantage could become an advantage for estimating influence for any query user (instead of a specific user) as the uniformly sampled vertices are generated independently of the query users. Based on this idea, we design the *reverse reachable sample graph* (or RR-GRAPH for simplicity) structure.

DEFINITION 2. *(*RR-GRAPH*) Given a social network $G = (V, E)$ and a vertex v,* RR-GRAPH *of v, denoted by $G_v^{\text{RR}} = (V(v), E(v))$, is sampled by: 1) Generate a random number $c(e) \in [0, 1]$ for all $e \in E$; 2) $V(v) \subseteq V$ contains vertices that reach v in G after removing all edges s.t. $p(e) < c(e)$, where $p(e) = \max_{i=1}^{Z} p(e|z_i)$; 3) $E(v) \subseteq V$ contains edges with both ends in $V(v)$ and associated with their $c(e)$.*

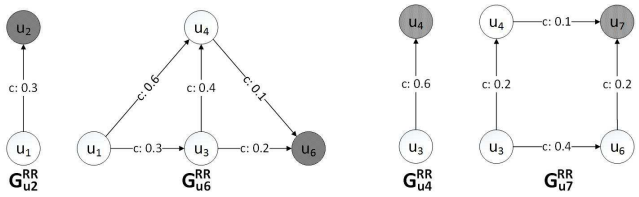


Figure 5: Example of RR-Graphs.

We can see that G_v^{RR} can be used as an instance of sample for any user and tag set in RR sampling due to the following reasons. First, as $p(e) = \max_{i=1}^{Z} p(e|z_i) \geq p(e|W)$ for any W, it will not miss any vertex that would influence v. Second, given any W, $c(e)$ can be used to examine if removing edge e by simply checking if $c(e) > p(e|W)$. In such a way, a RR-GRAPH generated can guarantee no vertex is missed from the corresponding RR iteration, and thus assure influence spread is not underestimated.

Next, we introduce a key operation on RR-GRAPH, namely tag-aware reachability, which is defined as follows.

DEFINITION 3. *(Tag-Aware Reachability in* RR-GRAPH*) Given a tag set W and a* RR-GRAPH *G_v^{RR}, a vertex u is said to reach v, denoted by indicator function $\mathbf{1}[u \rightsquigarrow v \mid G_v^{\text{RR}}, W] = 1$, if there exists at least one path from u to v in G_v^{RR} such that all edges in the path satisfy $p(e|W) \geq c(e)$.*

EXAMPLE 5. *Consider the running example in Fig. 2. Fig. 5 shows four* RR-GRAPH*s of the social graph, i.e. $G_{u_2}^{\text{RR}}, G_{u_6}^{\text{RR}}, G_{u_3}^{\text{RR}}$ and $G_{u_7}^{\text{RR}}$, with the random numbers associated with their edges. Given tag set $W = \{w_3, w_4\}$, we can see u_1 cannot reach u_2 in $G_{u_2}^{\text{RR}}$ since $p(u_1 \to u_2|W) = 0.13 < c(u_1 \to u_2) = 0.3$. In contrast, u_1 reaches u_6 in $G_{u_6}^{\text{RR}}$ via $u_1 \to u_3 \to u_4 \to u_6$ as $p(e|W) \geq c(e)$ for all edges in the path.*

Now, we are ready to present the index-based influence estimation, which is described in Algo. 3. For offline sampling, it generates a sufficient amount θ (determination of θ will be discussed later) of RR-GRAPHs for randomly picked vertices. For online estimation, it simply checks tag-aware reachability from u to v in each RR-GRAPH G_v^{RR}, and estimates influence as $\frac{\sum_{i=1}^{\theta} \mathbf{1}[u\rightsquigarrow v_i|G_{v_i}^{\text{RR}}, W]}{\theta} \cdot |V|$. This estimation is much more efficient than previous methods, as the costly sampling process is avoided and the RR-GRAPHs are usually much smaller than the original graph G. Moreover, as the samples are generated offline, we only process RR-GRAPHs containing u since u cannot reach the others.

EXAMPLE 6. *To estimate the influence of u_3 for the tag set $W = \{w_3, w_4\}$, the index-based method first examines* RR-GRAPH*s containing u_3, i.e., $\{G_{u_6}^{\text{RR}}, G_{u_4}^{\text{RR}}, G_{u_7}^{\text{RR}}\}$. As we only have $\mathbf{1}[u_3 \rightsquigarrow u_6 \mid G_{u_6}^{\text{RR}}, W]$ and $\mathbf{1}[u_3 \rightsquigarrow u_7 \mid G_{u_7}^{\text{RR}}, W]$ being 1, the influence can be estimated by $\frac{2}{4} \cdot 7 = 3.5$.*

Determination of Offline Sample Size. To guarantee the theoretical bounds mentioned previously, we need to make sure a sufficient amount θ of RR-GRAPHs are sampled offline. To this end, we show that whenever θ satisfies

$$\theta = \frac{2+\varepsilon}{\varepsilon^2} \cdot |V| \cdot (log(\delta) + log\phi_K + log2)\,, \tag{7}$$

Algorithm 3: EstimateInfluence+ $(u, G, W, \varepsilon, \delta)$

// Offline - RR-Graph sampling
Compute the offline sample size θ ;
for $i = 1, ..., \theta$ **do**
  Sample a random vertex v_i from V ;
  $G^{\mathtt{RR}}_{v_i} \leftarrow$ GenerateRRGraph (G, v_i);
// Online - RR-Graph matching
for $\forall G^{\mathtt{RR}}_{v_i}$ *s.t.* $u \in G^{\mathtt{RR}}_{v_i}$ **do**
  $\mathbf{1}[u \rightsquigarrow v_i \mid G^{\mathtt{RR}}_{v_i}, W] \leftarrow$ IsReachable $(u, v_i, G^{\mathtt{RR}}_{v_i}, W)$;
$\widehat{\mathbb{E}}[\mathcal{I}(u|W)] \leftarrow \frac{\sum_{i=1}^{\theta} \mathbf{1}[u \rightsquigarrow v_i | G^{\mathtt{RR}}_{v_i}, W]}{\theta} \cdot |V|$;
return $\widehat{\mathbb{E}}[\mathcal{I}(u|W)]$

where K is a parameter such that K is an upper bound of any possible k[2] and $\phi_K = \sum_{i \in [1,K]} \binom{|\Omega|}{i}$, the index-based influence estimation has a $\frac{1-\varepsilon}{1+\varepsilon}$ approximation ratio to the optimal solution with a $1 - \delta^{-1}$ probability (which can be proved similarly as Theorem 2).

Complexity Analysis. To study the time complexity of EstimateInfluence+, we have the following lemma.

Lemma 9. *For any u and W, the expected time to estimate $\mathbb{E}[\mathcal{I}(u|W)]$ using Algo. 3 is $O(\mathbb{E}^2[\mathcal{I}(u|*)]\frac{log(\delta)+log\phi_K+log2}{\varepsilon^2})$ where $\mathbb{E}[\mathcal{I}(u|*)]$ is the influence spread of u on a social graph with probability of each edge being $p(e) = \max_{i=1}^{Z} p(e|z_i)$.*

The term $\mathbb{E}^2[\mathcal{I}(u|*)]$ is often small due to the power law principle of social networks [9], when u is a randomly selected user. Thus, Algo. 3 can be much more efficient than previous influence estimation methods using online sampling. However, $\mathbb{E}^2[\mathcal{I}(u|*)]$ of an influential user could still be significantly large to prevent efficient query processing.

6.2 Effective Pruning Techniques

In this section, we present effective pruning techniques to further speed up the checking of tag-aware reachability in RR-Graphs. i.e., IsReachable in Algo. 3. Note that existing graph reachability techniques [17, 38] cannot be applied as they assume static graph structure or only allow incremental insertion/deletion of vertices/edges. However, each edge in an RR-Graph exists only when its influence probability $p(e|W) \geq c(e)$ for a given W. Thus the graph structures may be significantly vary for different tag sets.

To address this issue, we develop a filter-and-verification approach. Given a user u and a tag set W, the filter step prunes the *unpromising* RR-Graphs and generates a candidate set. Then, the verification step checks reachability for every candidate RR-Graph $G^{\mathtt{RR}}_v$ by removing the edges of $G^{\mathtt{RR}}_v$ s.t. $p(e|W) < c(e)$ and performing breath-first search from u in the resulted graph. This section focuses on presenting effective pruning techniques in the filter step.

The idea of our pruning technique is inspired by the *edge cut* in graph connectivity. Given a user u and an RR-Graph $G^{\mathtt{RR}}_v$, we select a set of edges such that u reaches v in $G^{\mathtt{RR}}_v$ given tag set W only if there exists at least one edge in the selected set satisfying $p(e|W) \geq c(e)$. Thus, instead of traversing $G^{\mathtt{RR}}_v$, we can prune $G^{\mathtt{RR}}_v$ if all selected edges meet $p(e|W) < c(e)$. Obviously, the quality of the selected edge cut determines the pruning effectiveness. Intuitively, we want to select a set of edges which are close to inactivated status, i.e. $p(e)$ is slightly larger than $c(e)$ for an edge e in a cut. Since $p(e|W) \leq p(e)$, the cut could easily prune corresponding RR-Graph if $p(e|W) < c(e)$.

[2]In PITEX, k will not be too large as any content of a social user containing too many tags will not be practical. In our experiments, we set $K = 10$.

Algorithm 4: RetainRRGraphs (u, G)

$G' = (V', E') \leftarrow$ a lazy sample from u on G (Algo. 17)
$v \leftarrow$ a uniform sample from V'
$V(v) \leftarrow \{v' | v' \in V' \wedge (v' \rightsquigarrow v)$ on $G'\}$
$E(v) \leftarrow \{(v'', v') | (v'', v') \in E' \wedge v'', v' \in V(v))$
$c(e) \leftarrow$ a uniform sample in $[0, p(e))$ $\forall e \in eset(v)$
return $G^{\mathtt{RR}}_v = (V(v), E(v))$

The edge cut construction is essentially an s-t minimum cut problem. Although the problem has been extensively studied [11, 3, 15], existing methods take at least a quadratic time (w.r.t. input graph size) to obtain merely an approximate solution on a graph with non-integral weights. To efficiently construct the cut, we apply a simple yet effective approach. By comparing two cuts: $E_{\mathtt{cut}}(u|v)'$ and $E_{\mathtt{cut}}(u|v)''$, where $E_{\mathtt{cut}}(u|v)'$ consists of u's out-going edges since u's neighbor can reach v and $E_{\mathtt{cut}}(u|v)''$ consists of edges from v's incoming neighbors, who can be reached by u, to v, we take the cut with a higher chance to prune as the filter.

Example 7. *Consider* RR-Graph $G^{\mathtt{RR}}_{u_7}$ *in Fig. 5. We compare two cuts:* $E_{\mathtt{cut}}(u_3|u_7)' = \{e_1 = (u_3, u_4), e_2 = (u_3, u_6)\}$ *and* $E_{\mathtt{cut}}(u_3|u_7)'' = \{e_3 = (u_4, u_7), e_4 = (u_6, u_7)\}$. *By assuming values of $p(e|W)$ are independent uniform r.v.s in $[0, p(e)]$, we can deduce the probabilities of the two cuts to prune $G^{\mathtt{RR}}_{u_7}$: 0.2 and 0.1 respectively. Thus we take $E_{\mathtt{cut}}(u_3|u_7)'$ as the edge cut for $G^{\mathtt{RR}}_{u_7}$.*

To enable fast pruning on top of the cut constructed, we employ an *inverted index* structure. Given a user u in a PITEX query, we first select the subset $\mathcal{G}^{\mathtt{RR}}(u)$ of the RR-Graphs that contain u, and compute edge cut $E_{\mathtt{cut}}(u|v)$ for each $G^{\mathtt{RR}}_v \in \mathcal{G}^{\mathtt{RR}}(u)$. Then, we construct inverted lists that map edges to RR-Graphs as follows. An inverted list of edge e contains a sorted set of RR-Graphs satisfying $e \in E_{\mathtt{cut}}(u|v)$ where the RR-Graphs are sorted by the $c(e)$ of e in ascending order. These inverted lists can be used to facilitate pruning for various tag sets. Specifically, given a tag set W, we examine each edge in the inverted lists. For each edge, we simply compute $p(e|W)$, scan the inverted list of e until $p(e|W) < c(e)$, and include all the visited RR-Graphs into the candidate set of the filter step. It is not difficult to show that all the unvisited RR-Graphs can be safely pruned without any additional computation.

Example 8. *Consider the* RR-Graphs *in Fig. 5 and a query user u_3. To answer a* PITEX *query of u_3, we first select the* RR-Graphs *containing u_3, i.e.,* $\{G^{\mathtt{RR}}_{u_6}, G^{\mathtt{RR}}_{u_4}, G^{\mathtt{RR}}_{u_7}\}$. *Then, we compute edge cuts and construct the inverted lists. Consider two edges, $e_1 = (u_3, u_4)$ and $e_2 = (u_3, u_6)$. The inverted lists are* $e_1 \rightarrow \{\langle G^{\mathtt{RR}}_{u_7}, 0.2\rangle, \langle G^{\mathtt{RR}}_{u_6}, 0.4\rangle, \langle G^{\mathtt{RR}}_{u_4}, 0.6\rangle\}$ *and* $e_2 \rightarrow \{\langle G^{\mathtt{RR}}_{u_6}, 0.2\rangle, \langle G^{\mathtt{RR}}_{u_7}, 0.4\rangle\}$. *Given a tag set $W = \{w_1, w_2\}$, we compute $p(e_1|W) = 0$ and $p(e_2|W) = 0.25$. Then, we can skip the inverted list of e_1 as all $c(e)$ in the list is larger than $p(e_1|W)$. Similarly, for the inverted list of e_2, we only need to visit $G^{\mathtt{RR}}_{u_6}$. Finally, we obtain a* RR-Graph *candidate set* $\{G^{\mathtt{RR}}_{u_6}\}$. *We can see that only one out of five* RR-Graphs *needs to be accessed, while the remaining are safely pruned.*

6.3 RR-Graphs Delay Materialization

To ensure the theoretical guarantee of the RR-Graphs based scheme, a large number of RR-Graphs instances are

Table 2: Statistics of Datasets

Datasets	$\|V\|$	$\|E\|$	$\|E\|/\|V\|$	$\|Z\|$	$\|\Omega\|$
lastfm	1.3K	12K	8.7	20	50
diggs	15K	0.2M	19.9	20	50
dblp	0.5M	6M	11.9	9	276
twitter	10M	12M	1.2	50	250

pre-computed and materialized. However, the size of all RR-Graph instances might be too large to fit into the memory for query processing on large graphs. To resolve this issue, we develop a delay materialization approach. The idea is to avoid storing the RR-Graphs during the index phase and only record how many RR-Graphs contain a user u, i.e. $\theta(u)$, for all $u \in V$. Whenever a user u issues a query, we "recover" $\theta(u)$ RR-Graphs and the rest of the processing remains the same as previously described in this section.

Example 9. *The delay materialization scheme maintains seven entries for the* RR-Graphs *shown in Fig. 5.* $\theta(u_1) = 2$, $\theta(u_2) = 1$, $\theta(u_3) = 3$, $\theta(u_4) = 3$, $\theta(u_5) = 0$, $\theta(u_6) = 2$, $\theta(u_7) = 1$. *If* u_3 *issues a query, after generating three* RR-Graphs, *e.g.* $\{G^{\mathtt{RR}}_{u_6}, G^{\mathtt{RR}}_{u_4}, G^{\mathtt{RR}}_{u_7}\}$, *the processing continues as illustrated in Example 8.*

Since the delay materialization scheme does not store any RR-Graph, to maintain the theoretical guarantee, it is crucial to ensure the samples generated during the query phase retain the same random distributions of the RR-Graphs initially constructed offline. Note that we cannot simply generate RR-Graphs by its definition since the naive scheme would likely generate RR-Graphs which exclude the query user u. This contradicts the fact that all RR-Graphs which need to be recovered in the query phase must contain u. There are two major issues for recovering the RR-Graphs: 1) the probability to sample any RR-Graph structure in the query phase must be equal to the one sampling the same RR-Graph structure which contains the query user in the index phase; 2) the random value $c(e)$ generated on each edge e of the recovered RR-Graphs must have the same distribution with that on the edges of the RR-Graphs originally generated offline.

For issue 1), a subgraph $G' = (V', E')$ is first extracted by a lazy sample g (Sec. 5.1) from u on G, where V' are the activated vertices in g and E' are the live edges in g. Then, we uniformly sample a vertex $v' \in V'$ and recover a RR-Graph as $G^{\mathtt{RR}}_{v'} = (V(v'), E(v'))$, where all vertices in V' reach v on G' and all edges have their both ends in V'. For issue 2), since the lazy sample (Algo. 2) does not assign a Bernoulli r.v. to each edge to decide if the edge is live or not, we assign a random value $c(e) \in [0, p(e))$ to each edge of the recovered RR-Graph for further influence estimations. The detailed procedure is presented in Algo. 4. The following theorem ensures the correctness for the recovering scheme with its proof established in Appx. B.10.

Theorem 3. *Using* RR-Graphs *recovered by Algo. 4 to estimate the influence for any user* u *and tag set* W *has the same approximation guarantee as that estimated by Algo. 3.*

7. EXPERIMENTS

This section evaluates the performance of our approaches for solving PITEX. First, we examine the empirical convergence of the sampling-based framework. Then, we compare the approaches in various parameter settings. Finally, we evaluate the scalability and conduct a case study.

7.1 Experimental Setup

Datasets. We conduct experiments on the following four real datasets.1) lastfm is a social music sharing dataset from an online site[3]. lastfm contains a social network and an action log which records users' activities of voting items (i.e., "a log of past propagation" in [2]). 2) diggs is an open social news dataset[4]. Like lastfm, it also contains a social network and an action log. 3) dblp is a DBLP co-author graph which is downloaded from an online academic search service[5]. 4) twitter is a social network built from the re-tweet and reply actions of users in Twitter. The dataset is downloaded from SNAP[6]. We adopt the TIC model [2] to compute the key probabilities $p(e|z)$ and $p(w|z)$ (see Section 3) in our model for lastfm and diggs based on their action logs. Following previous settings [2, 16, 6], we set the number of topics of these two datasets as 20. We also select top-50 frequent items (i.e., music/news) in these datasets to form tag set Ω. Since dblp has no action log, we follow the settings in [6] to use research fields as topics and compute $p(e|z)$ of two authors by categorizing their related conferences using the topics. We also use the keywords in the names of the conferences to formulate tag set Ω. For twitter dataset, we consider all hashtags of an individual user as a document and apply LDA [4] on all the documents to obtain the topic distribution of each user. Subsequently, we select the top-250 hashtags from re-tweets and replies in the edges as the tag set Ω. Given an edge $e = (u, v)$, we compute $p(e|z)$ based on the topic distributions of u and v. The statistics of these four datasets are listed in Table 2.

Query set and parameters. To evaluate the performance, we filter users with no outgoing edge and divide the rest of the users into three groups based on their out-degrees: high (top 1%), mid (top 1-10%) and low (the rest) with "mid" as the default group. For each user group, we generate 100 PITEX queries with randomly selected users within the group and average the results of the queries. We also discuss effect of parameters, including ε, δ and k, in Sec. 7.3.

Compared Approaches. We evaluate our best-effort exploration (Sec. 5.2) with the proposed strategies: lazy propagation sampling (Lazy), index-based estimation (IndexEst), index-based estimation with pruning (IndexEst+) and delay index materialization (DelayMat). Moreover, although we are the first to study PITEX, we can adapt the techniques of influence estimation in the state-of-the-art approaches to IM. Specifically, we compare with the following approaches: 1) sampling-based approaches MC [36] and RR [35] respectively employ Monte Carlo and Reverse Reachable Set sampling techniques for estimating influence (see Sec. 4). 2) tree-based approach Tim [6] utilizes a tree-based model to approximate the influence and develops bound estimation techniques to speedup influence computation.

We implement the above approaches, and integrate them into our framework for comparison.

Index Sizes and Time. We report the index sizes and construction time for all offline methods in Table 3. The size of RR-Graphs index is much larger than the original data size, since millions of RR-Graphs are stored in order to answer queries with accuracy guarantee. DelayMat (Sec. 6.3)

[3]http://www.last.fm/

[4]http://www.isi.edu/ lerman/downloads/digg2009.html

[5]http://dblp.uni-trier.de/xml/

[6]http://snap.stanford.edu/

Table 3: Index Sizes (MB) & Construction Time (s)

Datasets	Data	RR-Graphs		DelayMat	
		size	time	size	time
lastfm	0.86	6.02	0.19	0.005	0.32
diggs	20.8	58.4	9.17	0.07	7.09
dblp	210	5455	406	2.12	277
twitter	328	2912	276	20.9	184

drastically reduces the index sizes as it only keeps one entry for each user to record how many RR-Graphs contain the user. Moreover, DelayMat has a faster construction time than RR-Graphs index since it does not need to physically store the RR-Graph instances.

Experiment Settings. All the methods are implemented with C++ and ran on a CentOS server (Intel i7-3820 3.6GHz CPU with 8 cores and 60GB RAM).

7.2 Evaluation of Sampling Convergence

We first evaluate the empirical convergence of the proposed sampling-based framework. For each dataset, we consider the user with the largest out-degrees and its most influential tag as the tag set W. Then, we vary θ_W and use MC/RR/Lazy sampling strategy to estimate the influence.

Results in Fig. 6 show an interesting pattern. Recall that we have proved that both MC/RR/Lazy sampling strategies need the same sample size θ_W to guarantee the same error bound (see Lemmas 2 and 3). However, as observed from Fig. 6, MC/Lazy sampling converges faster than RR sampling: the former requires a smaller sample size than the latter for influence estimation. This can be explained by their estimation mechanisms. MC/Lazy uses θ_W i.i.d. $[0, 1]$ r.v.s while RR uses θ_W Bernoulli r.v.s for influence estimation. Ensuring theoretical bound of both methods requires the chernoff-hoeffding inequality, and Bernoulli r.v.s is the worst case for the chernoff-hoeffding inequality to hold [8]. Thus, RR converges slower than MC/Lazy.

7.3 Comparison of Approaches

In this section, we compare the approaches proposed in this paper by varying parameters.

Varying user group. We first vary user groups within which queries are generated from. Other parameters, i.e., ε, δ, k, are fixed to the default values: 0.7, 1000, 3 respectively, and the defaults are used throughout this section unless otherwise specified. We examine the efficiency of computing the most influential tag set as well as its influence spread.

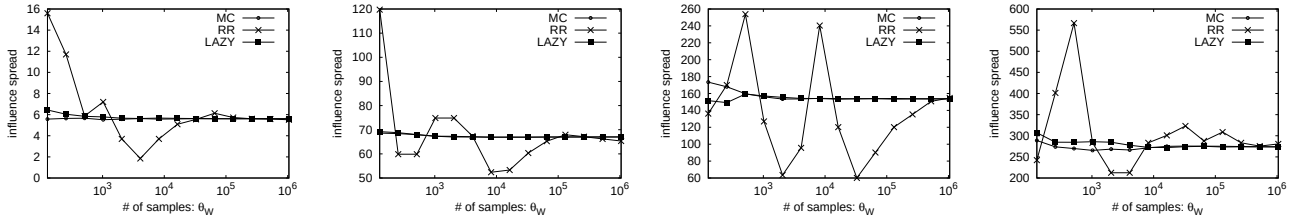
Fig. 7 shows the results of efficiency comparison. Among the online sampling methods, Lazy shows superior performance over MC/RR on all the datasets, which validates our claims in Sec. 4. We also compare the number of edges visited for online sampling methods (which is a measure of complexity) in Appx. D to further validate this finding. The index-based approach IndexEst significantly outperforms the online sampling methods. Take the dblp dataset as an example: IndexEst achieves 1031x, 504.7x, and 1562x speedups for the high, mid and low user group respectively. This is because the costly sampling process is avoided and influence can be estimated using the pre-computed RR-Graphs in IndexEst. On top of that, IndexEst+ further improves the efficiency and outperforms IndexEst by 4.9x, 6.4x, 4.7x for different user groups respectively, which shows the effectiveness of the edge-cut based pruning techniques. DelayMat runs slightly slower than IndexEst+ (but still performs better than IndexEst), as it has to "recover" RR-Graphs on-the-fly. However, we use it to trade significant reduction on index sizes as indicated in Table 3. Tim achieves better efficiency than the online sampling methods Lazy, MC and RR. That is because Tim utilizes a simplified tree-based method for estimating influence over graphs. However, this tree-based method does not have theoretical guarantee on influence spread, and thus may produce weak performance on influence spread, which will be discussed later. On the other hand, our index-based approaches, IndexEst, IndexEst+ and DelayMat, still perform much better than Tim on most of the datasets. For example, IndexEst+ and DelayMat are faster than Tim by at least three orders of magnitudes on the dblp dataset. This is because Tim has to examine many tag sets and generates the corresponding graphs for influence estimation in an online fashion. Instead, our approaches construct novel index RR-Graphs and pruning rules to speedup the computation. Note that the margin of our approach to Tim on the lastfm is not as much as other datasets. The reason is Tim only performs shortest path search to a limited number of vertices for influence estimation on smaller graphs, but are not scalable towards larger graph as shown in our experiments.

Fig. 8 compares the approaches on influence spread. Tim shows inferior performance compared with other approaches, since it employs a simplified tree-based model for estimating influence over graphs and does not have theoretical guarantee on influence spread. The approaches proposed in this paper have comparable influence spread in most of the cases, because all returned results are within the $\frac{1-\varepsilon}{1+\varepsilon}$ approximate ratio as stated in Theorem 2. We also note that the difference in the results generated from the various approaches is slightly larger on dblp and twitter datasets than that on the lastfm and diggs datasets. The reason is that the variance of random samples generated from larger social networks is higher, which leads to more volatile results.

As Lazy always shows the best performance among all online sampling methods and the influence scores returned are also within the error bound, we only compare Lazy with other offline solutions in the remaining part of this section.

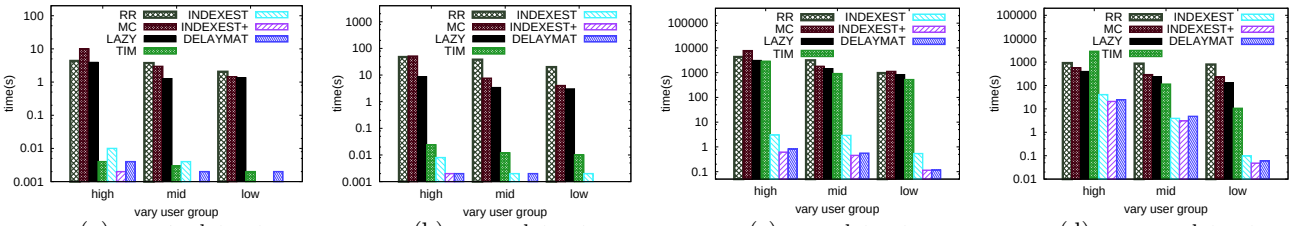
Varying parameter ε. We vary ε from 0.3 to 0.9. As shown in Fig. 9, the running time of all methods drops with a smaller ε as a larger ϵ leads to fewer samples being generated (by Lemma 2 and 3). Similar to the results when varying user groups, IndexEst shows its dominating performance over the online Lazy sampling method, e.g. respectively achieving speedups up to 712x, 2417x, 849x, and 90x on the four datasets. The edge-cut pruning techniques of IndexEst+ further boost the performance from IndexEst by 5x, 4x, 4x and 2x on the datasets respectively. These results again demonstrate the effectiveness of the RR-Graph-based method and edge-cut based pruning techniques. In addition, DelayMat is as efficient as IndexEst+ but only requires to use much smaller index spaces.

Fig. 10 shows the influence spread achieved by different methods. We can see that the influence spreads of these methods are quite close when ε is small (e.g., $\varepsilon = 0.3$), and when ε is larger, the differences also become larger. Since a larger ϵ leads to fewer samples being generated, which in turn makes the influence estimation less accurate. Note that the fluctuations in the influence result reported across all datasets are similar when varying other parameters, e.g.,



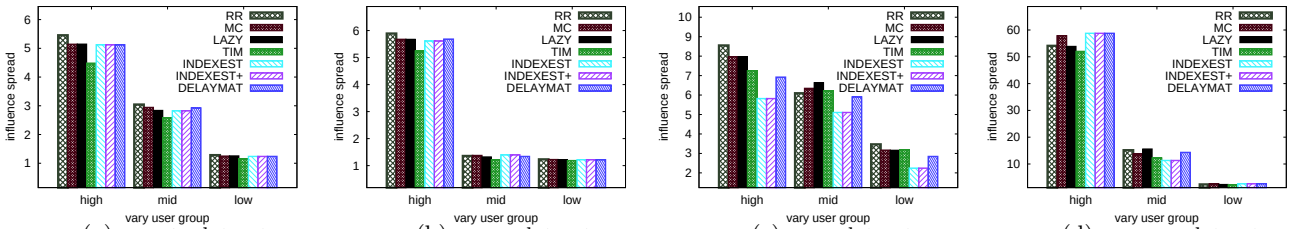
(a) `lastfm` dataset (b) `diggs` dataset (c) `dblp` dataset (d) `twitter` dataset

Figure 6: Evaluating empirical convergence of sampling-based influence estimation.



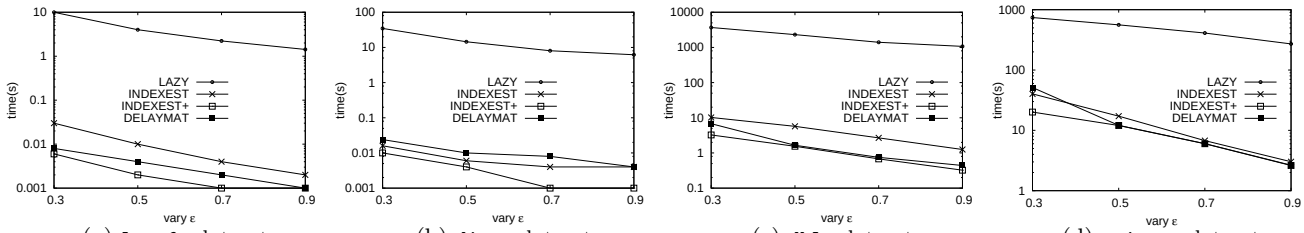
(a) `lastfm` dataset (b) `diggs` dataset (c) `dblp` dataset (d) `twitter` dataset

Figure 7: Efficiency comparison of methods when varying query user group.



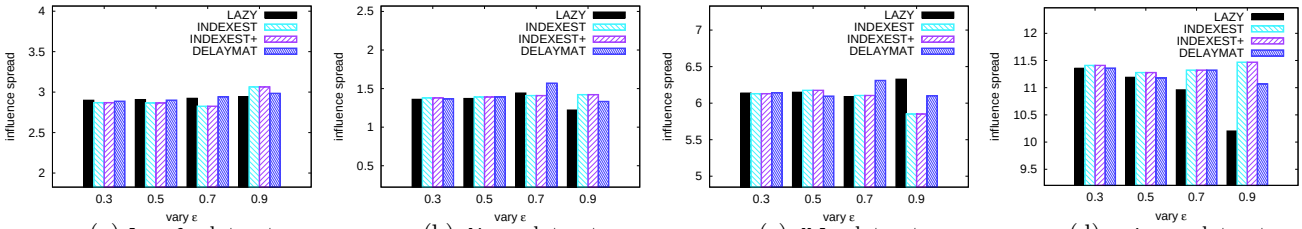
(a) `lastfm` dataset (b) `diggs` dataset (c) `dblp` dataset (d) `twitter` dataset

Figure 8: Influence spread comparison of methods when varying query user group.



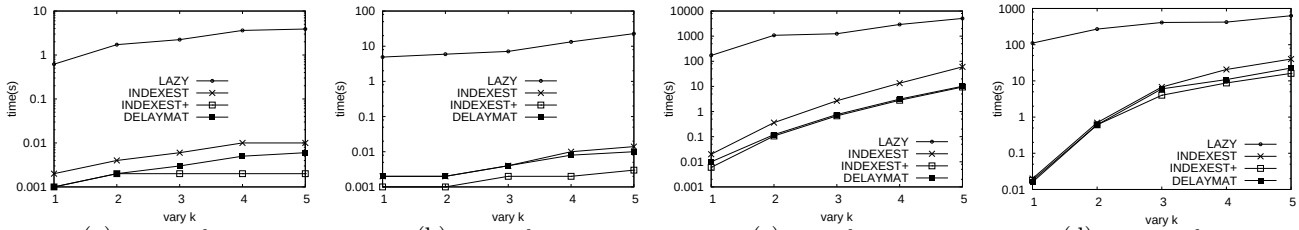
(a) `lastfm` dataset (b) `diggs` dataset (c) `dblp` dataset (d) `twitter` dataset

Figure 9: Efficiency comparison of methods when varying ε.



(a) `lastfm` dataset (b) `diggs` dataset (c) `dblp` dataset (d) `twitter` dataset

Figure 10: Influence spread comparison of methods when varying ε.

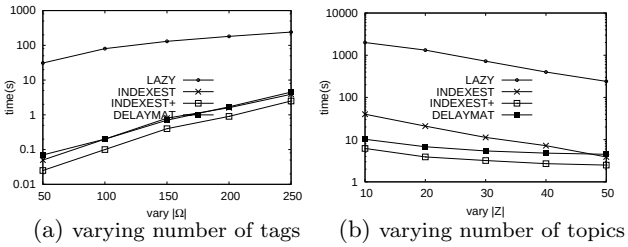


(a) `lastfm` dataset (b) `diggs` dataset (c) `dblp` dataset (d) `twitter` dataset

Figure 11: Efficiency comparison of methods when varying k.

Table 4: An example case study of PITEX query on `dblp` dataset.

Researchers	Inferential Tags	Accuracy	Researchers	Inferential Tags	Accuracy
Michael Jordan	systems, theory, speech learning, applications	0.80	Yann LeCun	image, recognition, theory, neural, representation	0.87
Jiawei Han	mining, structures, complexity optimization, programming	0.67	Jure Leskovec	society, communications, internet global, analysis	0.67
Michael Stonebraker	systems, distributed, dependable data, management	0.73	Jim Gray	parallel, distributed, theory, principles,storage	0.73
Richard Karp	mathematical, automata, combinatorial complexity, algorithms	0.93	Leslie Valiant	foundations, combinatorial, complexity, foundation, mathematical	0.87



(a) varying number of tags (b) varying number of topics

Figure 12: Scalability Evaluation (`twitter` dataset).

vary user group (Fig. 8), and thus we omit the results on influence spread in the remaining of this section.

As the results on the effect of parameter δ show similar trend, we leave its results and discussions in the appendix.

Varying parameter k. We examine the performance of all the methods by varying the tag number k.

We report the results in Fig. 11. First, when comparing the approaches, we have similar observations: IndexEst is at least two orders of magnitude better than Lazy, and IndexEst+ continues to be superior against IndexEst. Second, we also observe that the improvement achieved by IndexEst+ and DelayMat against IndexEst increases for larger k. This is because, when k is larger, IndexEst needs to check more tag sets and the pruning technique based on inverted index can enable more filtering power. Third, although the running time increases when increasing k in most cases, the running time of all the approaches does not explode exponentially w.r.t. increasing k despite an exponential growth in the number of k-size tag sets (Fig. 11). This is because the tag-topic probability densities[7] in most datasets are low, e.g., 0.16, 0.08, 0.32 and 0.17 for `lastfm`, `diggs`, `dblp` and `twitter` respectively. Note that when the density value is low, more tag sets tend to have zero probability against many topics and they are efficiently pruned by our best effort strategy (note that all the reported approaches adopt best effort exploration as mentioned previously).

7.4 Evaluation on Scalability

This section evaluates the scalability. We vary the number of tags $|\Omega|$ and the number of topics $|Z|$ on the largest `twitter` dataset. As shown in Fig. 12, with the increase of $|\Omega|$, although the running time increases for all the methods as the number of candidate tag sets becomes larger, IndexEst achieves the best performance against other methods and shows much better scalability against the number of tags. On the other hand, with the increase of $|Z|$, it is interesting to see that the running time decreases. The reason is as follows. When there are more topics, the tag-topic probability density becomes lower as each tag is often heavily distributed in one or few topics. As explained previously, lower tag-topic probability densities lead to better pruning abilities triggered by the best-effort strategy.

7.5 An Example Case Study

To illustrate the effectiveness, this section provides a case study on the co-authorship social graph `dblp`. We select 8 well-known computer scientists in various areas, machine learning (`Jordan` and `LeCun`), data mining (`Han` and `Leskovec`), databases (`Stonebraker` and `Gray`), and theory (`Karp` and `Lamport`). We generate PITEX queries with $k = 5$ for them, and conduct a real user survey for evaluate the effectiveness. We ask a group of PhD students with backgrounds in database management as annotators to manually judge the effectiveness of the system-recommended tags: given a target scientist, say `Han`, and a selected tag, say `mining`, the student will label 1 if she thinks the tag reflects the scientist's influential work, and 0 otherwise. For each query, we measure its *accuracy* by the ratio of 1's in the returned k tags, and average the accuracy from the all the students. Table 4 provides the selected tags as well as the survey results. We can see that the tags successfully summarize the "selling-points", such as "`distributed`", "`data`" and "`management`" of `Stonebraker`, and the average accuracy of all queries is 0.78. This example illustrates the potential usage of personalized social influential tags exploration in SNs[8].

8. CONCLUSION

In this paper, we introduced a new social influence problem, personalized social influential tags exploration (PITEX). We formalized this problem and proved the problem is NP hard to approximate within any constant ratio against the optimal solution. We developed a sampling-based framework and analyzed the drawbacks of two state-of-the-art sampling strategies for solving PITEX. To reduce the sampling complexities, we first proposed a lazy propagation sampling method to probe as fewer edges as possible and then employed a best-effort strategy to prune tag sets with insignificant influence spread. We further devised an index structure together with effective and materialization techniques to enable instant PITEX processing. We conducted extensive experiments on real large-scale social networks and the experimental results showed the effectiveness and efficiency of our methods.

Acknowledgment: This paper was partly supported by the NSFC under Grant No.61602488, No.61632016, No. 61602087 and No. 61632007, CCF-Tencent Open Research Fund (Project No. CCF-Tencent RAGR20160108), and the Start-up Research Grant of RUC (Project No. 16XNLF02).

[7]The tag-topic probability density is the ratio between the number of non-zero $p(w|z)$ entries and $|\Omega| \cdot |Z|$.

[8]In this paper, we focus on efficiency and will take more comprehensive user surveys for effectiveness evaluation in future work.

APPENDIX

A. REFERENCES

[1] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Trans. Knowl. Data Eng.*, 17(6), 2005.

[2] N. Barbieri, F. Bonchi, and G. Manco. Topic-aware social influence propagation models. In *ICDM*, 2012.

[3] A. A. Benczúr and D. R. Karger. Approximating s-t minimum cuts in $\tilde{O}$(n2) time. In *STOC*, pages 47–55, 1996.

[4] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. In *NIPS*, 2001.

[5] C. Borgs, M. Brautbar, J. T. Chayes, and B. Lucier. Influence maximization in social networks: Towards an optimal algorithmic solution. *CoRR*, 2012.

[6] S. Chen, J. Fan, G. Li, J. Feng, K. Tan, and J. Tang. Online topic-aware influence maximization. *PVLDB*, 8(6), 2015.

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

[8] F. Chung and L. Lu. Concentration inequalities and martingale inequalities: a survey. *Internet Math.*, 3(1):79–127, 2006.

[9] E. David and K. Jon. *Networks, Crowds, and Markets: Reasoning About a Highly Connected World*. Cambridge University Press, 2010.

[10] J. Fan, G. Li, and L. Zhou. Interactive SQL query suggestion: Making databases user-friendly. In *ICDE 2011*, pages 351–362, 2011.

[11] D. R. Ford and D. R. Fulkerson. *Flows in Networks*. Princeton University Press, 2010.

[12] F. Godin, V. Slavkovikj, W. De Neve, B. Schrauwen, and R. Van de Walle. Using topic models for twitter hashtag recommendation. In *WWW*, 2013.

[13] A. Goyal, F. Bonchi, and L. V. Lakshmanan. Learning influence probabilities in social networks. In *WSDM*, pages 241–250, 2010.

[14] M. Granovetter. Threshold models of collective behavior. *The American Journal of Sociology*, 83(6), 1978.

[15] J. Hao and J. B. Orlin. A faster algorithm for finding the minimum cut in a directed graph. *J. Algorithms*, 17(3):424–446, 1994.

[16] Çigdem Aslay, N. Barbieri, F. Bonchi, and R. A. Baeza-Yates. Online topic-aware influence maximization queries. In *EDBT*, 2014.

[17] R. Jin, H. Hong, H. Wang, N. Ruan, and Y. Xiang. Computing label-constraint reachability in graph databases. In *SIGMOD*, 2010.

[18] R. Jin, L. Liu, B. Ding, and H. Wang. Distance-constraint reachability computation in uncertain graphs. *Proc. VLDB Endow.*, 4(9):551–562, 2011.

[19] D. Kempe, J. Kleinberg, and E. Tardos. Maximizing the spread of influence through a social network. In *SIGKDD*, 2003.

[20] A. Khan, F. B. and1 Aristides Gionis, and F. Gullo. Fast reliability search in uncertain graphs. In *EDBT*, pages 535–546, 2014.

[21] J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. M. VanBriesen, and N. S. Glance. Cost-effective outbreak detection in networks. In *SIGKDD*, 2007.

[22] Y. Li, Z. Bao, G. Li, and K. Tan. Real time personalized search on social networks. In *ICDE*, pages 639–650, 2015.

[23] Y. Li, J. Fan, D. Zhang, and K.-L. Tan. Discovering your selling points: Personalized social influential tags exploration. Technical report, 2017. www.comp.nus.edu.sg/~a0047194/tr2016-pitr.pdf.

[24] Y. Li, D. Zhang, Z. Lan, and K. Tan. Context-aware advertisement recommendation for high-speed social news feeding. In *ICDE*, pages 505–516, 2016.

[25] Y. Li, D. Zhang, and K. Tan. Real-time targeted influence maximization for online advertisements. *PVLDB*, 8(10):1070–1081, 2015.

[26] L. Liu, J. Tang, J. Han, M. Jiang, and S. Yang. Mining topic-level influence in heterogeneous networks. In *CIKM*, 2010.

[27] K. Mao, J. Fan, L. Shou, G. Chen, and M. S. Kankanhalli. Song recommendation for social singing community. In *ACM MM*, pages 127–136, 2014.

[28] H. T. Nguyen, M. T. Thai, and T. N. Dinh. Stop-and-stare: Optimal sampling algorithms for viral marketing in billion-scale networks. In *SIGMOD*, pages 695–710, 2016.

[29] M. Potamias, F. Bonchi, A. Gionis, and G. Kollios. K-nearest neighbors in uncertain graphs. *Proc. VLDB Endow.*, 3(1-2):997–1008, 2010.

[30] M. Renz, R. Cheng, and H.-P. Kriegel. Similarity search and mining in uncertain databases. *Proc. VLDB Endow.*, 3(1-2):1653–1654, 2010.

[31] I. Ronen, I. Guy, E. Kravi, and M. Barnea. Recommending social media content to community owners. In *SIGIR*, 2014.

[32] S. Sedhai and A. Sun. Hashtag recommendation for hyperlinked tweets. In *SIGIR*, 2014.

[33] J. Tang, X. Hu, and H. Liu. Social recommendation: a review. *Social Netw. Analys. Mining*, 3(4), 2013.

[34] J. Tang, J. Sun, C. Wang, and Z. Yang. Social influence analysis in large-scale networks. In *KDD*, pages 807–816, 2009.

[35] Y. Tang, Y. Shi, and X. Xiao. Influence maximization in near-linear time: A martingale approach. In *SIGMOD*, pages 1539–1554, 2015.

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

[37] V. V. Vazirani. *Approximation algorithms*. Springer, 2001.

[38] K. Xu, L. Zou, J. X. Yu, L. Chen, Y. Xiao, and D. Zhao. Answering label-constraint reachability in large graphs. In *CIKM*, 2011.

B. THEOREMS AND PROOFS

B.1 Proof of Lemma 1

PROOF. First, it is easy to verify that the k-label s-t reachability problem is equivalent to the problem of finding the minimum number of labels required so that s reaches t. Next, we prove latter problem is NP-hard, by a reduction

from the set cover problem. Given an instance of the set cover problem with a universe set $U = \{u_1, ..., u_n\}$ and a collection of subsets $\mathcal{S} = \{S_1, S_2, ..., S_m\}, S_i \subseteq U$, we construct a directed multi-graph G' with $n+1$ vertices $\{u'_1, \ldots, u'_{n+1}\}$ and m labels $\{l_1, \ldots, l_m\}$. Then if $u_i \in S_j$, we add an edge (u'_i, u'_{i+1}) with a label w_j into G'. It is not difficult to see that the problem of finding the minimum labels that u'_1 reaches u'_{n+1} on G' can be solved, only if the set cover problem is solved. As the reduction is in polynomial time and the set cover problem is NP-hard, we finish the proof. □

B.2 Chernoff Bounds

Let X be the sum of θ independent and identical r.v.s sampled from a distribution on $[0, 1]$ with a mean p. For any $\delta > 0$, the followings hold [8],

$$\Pr[X - \theta p \geq \delta \cdot \theta p] \leq exp(-\frac{\delta^2}{2+\delta}),$$
$$\Pr[X - \theta p \leq -\delta \cdot \theta p] \leq exp(-\frac{\delta^2}{2})$$

B.3 Proof of Lemma 3

Proof. We state the following probability bound:

$$Pr\left[\left|\widehat{\mathbb{E}}_{\mathrm{MC}}[\mathcal{I}(u|W)] - \mathbb{E}[\mathcal{I}(u|W)]\right| \geq \varepsilon \cdot \mathbb{E}[\mathcal{I}(u|W)]\right] \quad (8)$$
$$= Pr\left[\left|\frac{\sum_{i=1}^{\theta_W} \mathcal{I}_{g_i}(u|W)}{|\mathcal{R}_W(u)|} - \frac{\theta_W \cdot \mathbb{E}[\mathcal{I}(u|W)]}{|\mathcal{R}_W(u)|}\right| \geq \varepsilon \cdot \frac{\theta_W \mathbb{E}[\mathcal{I}(u|W)]}{|\mathcal{R}_W(u)|}\right]$$

Note that $\frac{\sum_{i=1}^{\theta_W} \mathcal{I}_{g_i}(u|W)}{|\mathcal{R}_W(u)|}$ is a sum of θ_W i.i.d. r.v.s and each variable $\frac{\mathcal{I}_{g_i}(u|W)}{|\mathcal{R}_W(u)|}$ lies in $[0, 1]$. Moreover, we have $\mathbb{E}[\frac{\sum_{i=1}^{\theta_W} \mathcal{I}_{g_i}(u|W)}{\theta_W}] = \mathbb{E}[\mathcal{I}(u|W)]$. Thus, the following holds according to the chernoff bound [8]:

$$\text{Eqn. (8)} < 2 \cdot \exp\left(-\frac{\varepsilon^2}{2+\varepsilon} \cdot \theta_W \cdot \frac{\mathbb{E}[\mathcal{I}(u|W)]}{|\mathcal{R}_W(u)|}\right) = \frac{1}{\delta \cdot \binom{|\Omega|}{k}} \quad (9)$$

Therefore, Lemma 3 is proved. □

B.4 Proof of Lemma 5

Proof. Consider an instance of sample g in MC sampling. Let $\mathcal{R}^g_W(u)$ denote the set of vertices in g reachable from u, and p_g denote the probability of randomly selecting an edge from $E_W(u)$ that starts from a vertex in $\mathcal{R}^g_W(u)$. Then, we have $\texttt{ENE}^{\mathrm{MC}} = \mathbb{E}[|E_W(u)| \cdot p_g]$ where the expectation is taken over the randomness of $\mathcal{R}^g_W(u)$.

Let us choose a random vertex $v \in V$ with the probability proportional to its *out-degree*, denoted by $\deg(v)$, and let $\mathbf{1}[v \in \mathcal{R}^g_W(u)]$ be an indicator function such that $\mathbf{1}[v \in \mathcal{R}^g_W(u)]$ equals 1 if $v \in \mathcal{R}^g_W(u)$ and equals 0 otherwise. We have:

$$\frac{\texttt{ENE}^{\mathrm{MC}}}{|E_W(u)|} = \mathbb{E}[p_g] = \sum_g (\Pr[g] \cdot p_g)$$
$$= \sum_g \left(\Pr[g] \cdot \frac{\sum_v \deg(v) \cdot \mathbf{1}[v \in \mathcal{R}^g_W(u)]}{|E_W(u)|}\right)$$
$$= \sum_v \left(\frac{\deg(v)}{|E_W(u)|} \cdot \sum_g \left(\Pr[g] \cdot \mathbf{1}[v \in \mathcal{R}^g_W(u)]\right)\right)$$
$$= \sum_v \left(\frac{\deg(v)}{|E_W(u)|} \cdot \Pr[u \rightsquigarrow v]\right) = \mathbb{E}[\mathcal{I}(u \rightsquigarrow v^{ot}|W)]$$

which completes the proof. □

B.5 Proof of Theorem 2

Proof. Let W^* and $\widehat{W}$ respectively denote the optimal tag set and the tag set returned by the enumeration-based method. Recall that $\widehat{\mathbb{E}}[\mathcal{I}(u|W)]$ is the influence spread estimated by either MC or RR. Based on Lemmas 2 and 3, the estimation has an error bound for any k-size tag set W with a probability of at least $1 - \delta^{-1}\binom{\Omega}{k}^{-1}$ when θ_W is large. By union bound [37], the estimation has the error bound with a probability of at least $1 - \delta^{-1}$ simultaneously for all k-size tag sets. Thus, the following holds with a probability of $1 - \delta^{-1}$:

$$\mathbb{E}[\mathcal{I}(u|\widehat{W})] > \frac{\widehat{\mathbb{E}}[\mathcal{I}(u|\widehat{W})]}{1+\varepsilon} \geq \frac{\widehat{\mathbb{E}}[\mathcal{I}(u|W^*)]}{1+\varepsilon} \geq \frac{1-\varepsilon}{1+\varepsilon} \cdot \mathbb{E}[\mathcal{I}(u|W^*)]$$

which proves the theorem. □

B.6 Proof of Lemma 6

Let Y denote event 1), then $\Pr[Y = y] = \binom{\theta}{y} p^y (1-p)^{\theta-y}$, $\forall y \in 1..\theta$. Let Y' denote event 2) and X_is are i.i.d geometric r.v.s with probability p, we derive the followings:

$$\begin{aligned}
\Pr[Y' = y] &= \sum_{s=y..\theta} \Pr[\sum_{i=1..y} X_i = s] \cdot \Pr[X_{y+1} > \theta - s] \\
&= \sum_{s=y..\theta} \binom{s-1}{y-1} p^y (1-p)^{s-y} (1-p)^{\theta-s} \\
&= \left\{\binom{y}{y} + \sum_{s=y+1..\theta} \binom{s-1}{y-1}\right\} \cdot p^y (1-p)^{\theta-y} \\
&= \binom{\theta}{y} p^y (1-p)^{\theta-y} = \Pr[Y = y]
\end{aligned}$$

where the second equality is based on the fact the sum of y geometric random variable with success probability of p is equal to a negative binomial distribution with parameter y and p. Then it is safe to say two events are statistically identical since they have the same probability for all instances.

B.7 Proof of Lemma 7

Proof. Consider an instance of sample g in lazy propagation sampling. Let $\mathcal{R}^g_W(u)$ denote the set of vertices in g reachable from u, only edges which has both end vertices in $\mathcal{R}^g_W(u)$ are traversed. According to the IC model, the influence probability through any edge $(x \to y)$ is inverse proportional to the in-degree of y [19, 7, 36]. If $c(x)$ denote the number of vertices which are active and incident to x, $\texttt{ENE}^{\mathrm{LP}}$ is evaluated as the following:

$$\begin{aligned}
\texttt{ENE}^{\mathrm{LP}} &= \sum_g \Pr[g] \cdot \sum_{x \in \mathcal{R}^g_W(u)} c(x) \\
&= \sum_g \Pr[g] \cdot \sum_{x \in \mathcal{R}^g_W(u)} \frac{|\{(x,y) \in E | y \in \mathcal{R}^g_W(u)\}|}{deg^{in}(x)} \\
&\leq \sum_g \Pr[g] \cdot |\mathcal{R}^g_W(u)| \\
&\leq |\mathcal{R}_W(u)| \sum_g \Pr[g] \cdot \frac{|\mathcal{R}^g_W(u)|}{|\mathcal{R}_W(u)|} \\
&\leq |\mathcal{R}_W(u)| \cdot \mathbb{E}[\mathcal{I}(u \rightsquigarrow v^*|W)]
\end{aligned}$$

where v^* is uniformly selected from $\mathcal{R}_W(u)$. □

B.8 Proof of Lemma 8

PROOF. It is straightforward to see that:

$$p^+(e|W) \leq \max_{p(z|W)>0 \wedge z \in Z} p(e|Z)$$

Thus we are only left to prove: $p^+(e|W) \leq$ Eqn. 6. To achieve this, let $p_z^+(e|W) = \frac{\prod_{w\in W} p(w|z)p(z)}{\sum_{z'\in Z}\prod_{w\in W} p(w|z')p(z')}$, we have the following inequalities:

$$log(p_z^+(e|W)) = log\Big(\frac{\prod_{w\in W} p(w|z)p(z)}{\sum_{z'\in Z}\prod_{w\in W} p(w|z')p(z')}\Big)$$
$$= \sum_{w\in W} log(p(w|z)p(z)) - log\Big(\sum_{z'\in Z}\prod_{w\in W} p(w|z')p(z')\Big) \quad (10)$$

Note when $p(w|z) = 0$ for any $w \in W$, $p_z^+(e|W) = 0$ and no estimation is needed. Subsequently, we apply the jensen inequality to Eqn. 10.

$$log(p_z^+(e|W)) \leq \sum_{w\in W} log(p(w|z)p(z)) - \sum_{z'\in Z} p(z') \sum_{w\in W} log(p(w|z'))$$
$$\leq log(\prod_{w\in W} p(w|z)p(z)) - log(\prod_{w\in W}\prod_{z'\in Z} p(w|z')^{p(z')})$$
$$\leq log\Big(\prod_{w\in W}\frac{p(w|z)p(z)}{\prod_{z'\in Z} p(w|z')^{p(z')}}\Big) \quad (11)$$

As natural logarithm is a monotone increasing function, $p_z^+(e|W) \leq \prod_{w\in W}\frac{p(w|z)p(z)}{\prod_{z'\in Z} p(w|z')^{p(z')}}, \forall W \subset \Omega$. The rest is simply to plug in Eqn. 1 to complete the proof. □

B.9 Proof of Lemma 9

PROOF. As the influence of u estimated directly by RR-GRAPHs generated offline is computed as $\mathbb{E}[\mathcal{I}(u|*)] = \frac{r}{\theta}|V|$ where r is the number of RR-GRAPHs containing u. Thus $r = \mathbb{E}[\mathcal{I}(u|*)]\frac{\theta}{|V|}$. This means if the first $\theta' = \frac{\theta}{\mathbb{E}[\mathcal{I}(u|W)]}$ RR-GRAPHs are used to estimate $\mathbb{E}[\mathcal{I}(u|W)]$, the number of RR-GRAPHs containing u is $r' = \frac{r}{\mathbb{E}[\mathcal{I}(u|W)]}$. Moreover, according to Lemma 3, using θ' RR-GRAPHs for estimation can deliver an approximation of error ratio between $(1-\varepsilon, 1+\varepsilon)$ with probability of at least $1 - \delta^{-1}\binom{\Omega}{K}^{-1}$. It follows that the expected estimation complexity is computed by $r'\cdot$ENE where ENE is the expected number of visited edges to compute one RR-GRAPH.

Given a tag set W, for each RR-GRAPH $G_v^{\mathtt{RR}}$ containing u, we can perform a breadth first search (BFS) from u to check if u reaches v in $G_v^{\mathtt{RR}}$. Consider an instance of sample g. Let $\mathcal{R}^g(u)$ denote the set of vertices in g reachable from u and $\mathcal{R}^g_W(u)$ denote the set of vertices in g reachable from u after removing any edge e if $p(e|W) < c(e)$, we have $\mathtt{ENE} = \sum_g \Pr[g] \cdot \sum_{x\in\mathcal{R}^g_W(u)} |\{(x,y) \in E | y \in \mathcal{R}^g(u)\}|$. This is because we only visit the vertices in $\mathcal{R}^g_W(u)$ but probe edges which points to a vertex in $\mathcal{R}^g(u)$. Since $\sum_{x\in\mathcal{R}^g_W(u)} |\{(x,y) \in E | y \in \mathcal{R}^g(u)\}| \leq c\cdot\mathbb{E}[\mathcal{I}(u|W)]\cdot|\mathcal{R}^g(u)|$ for some c, we have

$$\mathtt{ENE} \leq O\Big(\mathbb{E}[\mathcal{I}(u|W)] \cdot \sum_g \Pr[g] \cdot |\mathcal{R}^g(u)|\Big)$$
$$\leq O\Big(\mathbb{E}[\mathcal{I}(u|*)]\mathbb{E}[\mathcal{I}(u|W)]\Big)$$

Given the above derivation, the expected estimation complexity is bounded by $r'\cdot\mathtt{ENE} \leq \frac{r}{\mathbb{E}[\mathcal{I}(u|W)]}\cdot\mathbb{E}^2[\mathcal{I}(u|*)]$. Finally, Lemma 9 is obtained by simplify the expression. □

Algorithm 5: BESTEFFORTSAMPLE $(u, G, k, \varepsilon, \delta)$

```
Initialize a max-heap H ;
W* ← ∅, I* ← 0 ;
Insert ⟨∅, 0⟩ into H ;
while H ≠ ∅ do
    ⟨W, I⟩ ← H.pop() ;
    if |W|=k then
        I ← EstimateInfluence (u, G, W, ε, δ) ;
        if I > I* then W* ← W, I* ← I ;
    else
        I ← EstimateUpperBound (u, G, W, ε, δ) ;
        if I ≤ I* then continue ;
        for w < w'  ∀w' ∈ W ∧ w ∈ Ω do
            W' ← W ∪ {w} ;
            Insert ⟨W', I⟩ into H ;
return ⟨W*, I*⟩ ;
```

B.10 Proof of Theorem 3

Let v^* to be a uniformly selected vertex from V and X to be the event that a particular RR-GRAPH of v^*, i.e. $G_{v^*}^{\mathtt{RR}}$ is sampled given the condition that $G_{v^*}^{\mathtt{RR}}$ contains the query user u. Moreover, let g denote a random subgraph of G by removing each edge with a probability of $1-p(e)$ and $R_g(u)$ denote the set of vertices reached by u on g with $R(u)$ as the random version of $R_g(u)$ by considering the randomness in g. We then derive the following equation:

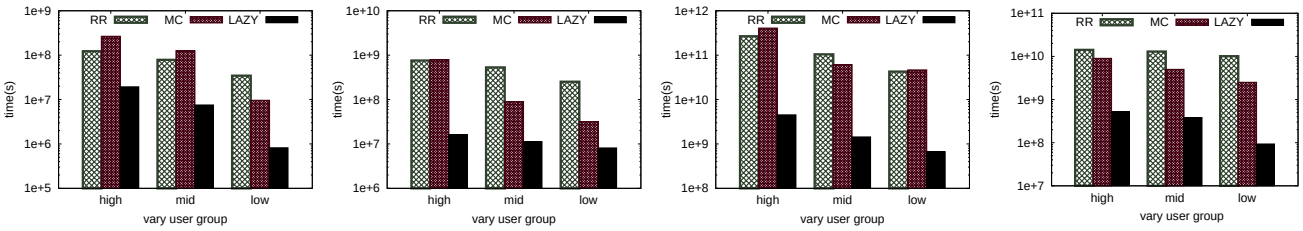
$$\Pr[X] = \Pr[G_{v^*}^{\mathtt{RR}}|u \in G_{v^*}^{\mathtt{RR}}] = \sum_g \Pr[g] \cdot \sum_{v^*\in V} \Pr[v^*] \cdot b_g(u \rightsquigarrow v^*)$$
$$= \Pr[g] \cdot \sum_{v'\in R_g(u)} \Pr[v'] \cdot b_g(u \rightsquigarrow v')$$
$$= \Pr[G_{v'}^{\mathtt{RR}}|v' \in R(u)]$$

where $b_g(u \rightsquigarrow v^*) = 1$ if u reaches v^* on g and 0 otherwise. Note that the resulting event is exactly the same as the procedure in Algo. 4.

At this point, we establish the statistical equivalence between pure index and delay sampling approaches in terms of generating graph structures. We has yet to prove the random values generated by pure index and delay sampling approaches share the same distribution. Let $c(e)$ denote the random values generated by the pure index approach, the edge being considered in a RR-GRAPH must be live, i.e. $c(e) < p(e)$. Given any tag set W and $p(e|W) < p(e)$, the following equality is easy to obtain: $\Pr[c(e) < p(e|W)|c(e) < p(e)] = \frac{\Pr[c(e)<p(e|W)]}{\Pr[c(e)<p(e)]} = \frac{p(e|W)}{p(e)}$. This is exactly the probability to generate a random value $c'(e) \in [0, p(e))$ which is live w.r.t. W, as presented in Algo. 4.

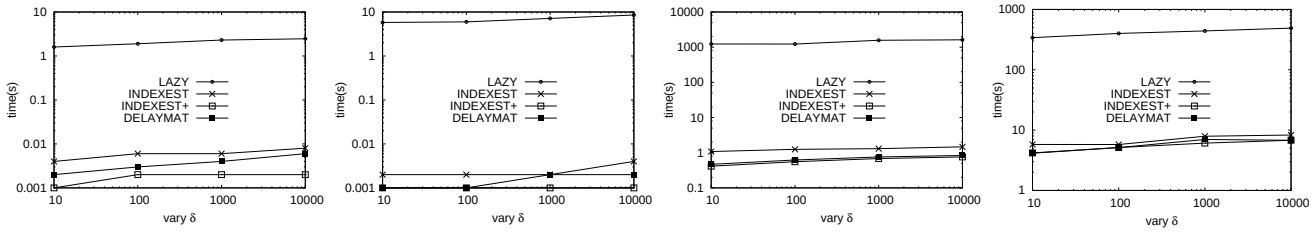
C. BEST EFFORT EXPLORATION

Algo. 5 provides the pseudo-code of this framework. Let Ω to be an ordered set where the order can be arbitrary, and we denote $w_i < w_j$ if $w_i, w_j \in \Omega$ and $i < j$. It employs a max-heap $\mathcal{H}$ to preferentially access (partial) solutions with larger upper bound of influence spread. Initially, it inserts an empty tag set $\langle\emptyset, 0\rangle$ into $\mathcal{H}$ and selects tag sets iteratively. In each iteration, it pops a partial solution W with its upper bound $\mathcal{I}$ from $\mathcal{H}$. If W already has k tags, it simply computes the estimated influence spread by applying the lazy propagation sampling to update the current best solution



(a) `lastfm` dataset (b) `diggs` dataset (c) `dblp` dataset (d) `twitter` dataset

Figure 13: Number of visited edges for online sampling methods when varying user groups.



(a) `lastfm` dataset (b) `diggs` dataset (c) `dblp` dataset (d) `twitter` dataset

Figure 14: Efficiency comparison of methods when varying δ.

(Lines 6-8). Otherwise where W has less than k tags, it first estimates the upper bound for W using EstimateUpperBound (Line 10) and prunes W if its upper bound is smaller or equal to the current best size-k solution. Lastly it adds one more tag w into W for forming W' and inserts W' back into $\mathcal{H}$. The algorithm terminates when $\mathcal{H}$ is empty and returns the best solution $\langle W^*, \mathcal{I}^* \rangle$. Complementary example can be found in [23].

Theoretical analysis: Note that the number of tag sets examined in the enumeration-based sampling is $\binom{|\Omega|}{k}$ whereas the number is $\sum_{i=1,..,k}\binom{|\Omega|}{k}$ for best-effort exploration in the worst case. To retain the same theoretical guarantee as stated in Theorem 2, the sample size θ_W required for best effort sampling w.r.t. any partial tag set W should satisfy:

$$\theta_W \geq \frac{2+\varepsilon}{\varepsilon^2} \cdot |\mathcal{R}_W(u)| \cdot \frac{log(\delta) + log\left[\sum_{i=1}^{k}\binom{|\Omega|}{i}\right] + log2}{\mathbb{E}[\mathcal{I}(u|W)]} \quad (12)$$

In the "worst" case, the best-effort strategy is required to estimate the influence for ϕ_k tag sets, where $\phi_k = \sum_{i\in[1,k]}\binom{|\Omega|}{i}$. However, we can derive that: $\phi_k \leq \binom{|\Omega|}{k}\frac{|\Omega|-k+1}{|\Omega|-2k+1}$. Since in practice $k \ll |\Omega|$ which leads $O(\phi_k) = O(\binom{|\Omega|}{k})$. Thus the "worst" complexity of best-effort sampling is the same as that of enumeration-based sampling.

D. ADDITIONAL EXPERIMENTAL RESULTS

Comparing edge visits for online sampling methods. In Fig. 13, we present the experimental results comparing online sampling methods in terms of number of edges visited during the influence estimation process. It is not surprising that processing PITEX queries for high degree users requires probing more edges compared to those for users with lower degrees. The reason behind such an observation is that high degree users tend to have influence over a large number of users w.r.t. many tag sets. We see that MC and RR can outperforms each other in different scenarios, as predicted by our analysis in Sec. 4. In fact, ratio of MC over RR for the number of edges probed is proportional to $\frac{\mathbb{E}[\mathcal{I}(u\rightsquigarrow v^{ot}|W)]}{\mathbb{E}[\mathcal{I}(v^{in}\rightsquigarrow v^{*}|W)]}$ (Lemma 4 and 5) and the ratio varies for different scenarios. Nevertheless, we find Lazy visits a much smaller number of edges compared to MC and RR. Due to the sparseness nature of the influence graph (Sec. 5), MC and RR performs redundant visits of edges which are indeed non-active. As Lazy only visits edges which are active, it results in more than an order of magnitude of improvement of Lazy over MC/RR for the number of probing edges across all datasets. However, we note that such huge improvements in edge visits does not fully translate to run time efficiency improvement as presented in Fig. 7. The main reason is, for any tag set under influence estimation, we need to create a priority queue for each visited user and delete all queues after the tag set computation. As there are millions of users in the social graph with hundreds of tag sets are evaluated, such intensive creation and deletion of the priority queues generate a large overheads to Lazy. To improve efficiency, one direction is to reuse the priority queues. We leave this as a future work.

Vary parameter δ. We vary δ for 10, 100, 1000 and 10000. In Fig. 14, we observe that, although the running time of all methods increases with a large δ, the performance does not become worse in an exponential fashion as δ explodes. Such a phenomenon is easy to interpret according to Eqn. 2, as the number of samples used to estimate the influence is proportional to $log(\delta)$. Similar to results shown earlier in Sec. 7, index-based approaches achieve orders of magnitude speedups compare to the online sampling methods, which once again justifies the effectiveness of using index-based methods.