Finding Seeds and Relevant Tags Jointly: For Targeted Influence Maximization in Social Networks

Xiangyu Ke
NTU Singapore
xiangyu001@e.ntu.edu.sg

Arijit Khan
NTU Singapore
arijit.khan@ntu.edu.sg

Gao Cong
NTU Singapore
gaocong@ntu.edu.sg

ABSTRACT

We study the novel problem of jointly finding the top-k seed nodes and the top-r relevant tags for targeted influence maximization in a social network. The bulk of the research on influence maximization assumes that the influence diffusion probabilities across edges are fixed, and the top-k seed users are identified to maximize the cascade in the entire graph. However, in real-world applications, edge probabilities typically depend on the information being cascaded, e.g., in social influence networks, the probability that a tweet of some user will be re-tweeted by her followers depends on whether the tweet contains specific hashtags. In addition, a campaigner often has a specific group of target customers in mind.

In this work, we model such practical constraints, and investigate the novel problem of jointly finding the top-k seed nodes and the top-r relevant tags that maximize the influence inside a target set of users. Due to the hardness of the influence maximization problem, we develop heuristic solutions — with smart indexing, iterative algorithms, and good initial conditions, which target high-quality, efficiency, and scalability.

CCS CONCEPTS

• **Information systems** → **Social advertising**; **Social networks**;
• **Theory of computation** → **Sketching and sampling**;

KEYWORDS

Targeted Influence Maximization; Reverse Sketching; Indexing; Conditional Influence Probability

ACM Reference Format:
Xiangyu Ke, Arijit Khan, and Gao Cong. 2018. Finding Seeds and Relevant Tags Jointly: For Targeted Influence Maximization in Social Networks. In *SIGMOD'18: 2018 International Conference on Management of Data, June 10–15, 2018, Houston, TX, USA.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3183713.3199670

1 INTRODUCTION

The classical influence maximization problem [7, 12] identifies the top-k seed users in a social network such that the expected number of influenced users in the network, starting from those seeds and following an influence diffusion model, is maximized. The budget k on the seed set size usually depends on the campaigner, e.g., it may depend on how many initial users the campaigner can directly influence by advertisements, free samples, and discounted prices.

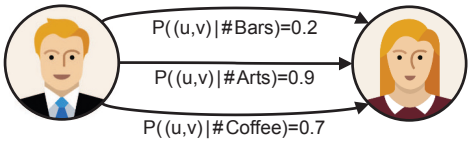


Figure 1: Edge influence probabilities based on information topics

In this paper, we consider a practical scenario where the campaigner additionally specifies her target customers, that is, we employ the notion of targeted influence maximization [15, 17, 21, 27]. It is easy for a campaigner to define her target users, either explicitly, or via some constraints, e.g., people in the age group 20-30, all IT professionals in the silicon valley, etc. The number of such target customers can be very large, and it is often not possible (or not economical) to reach out to everyone by advertisements, giving free samples, or discounted price. Therefore, the campaigner still prefers to find a small set of seed nodes (i.e., top-k) in order to maximize the spread of her campaign within these target users.

Many problem variants of influence maximization have been considered in the literature, majority of them assuming that the influence cascade probabilities between two users are fixed and without taking into consideration the actual information being cascaded. Users generally apply tags to characterize their contents in an online social network, e.g., hashtags in *Twitter* and *Instagram*. Moreover, one can identify representative keywords (e.g., the most frequent ones after removing stop words) from the contents, and use them as tags. The probability that a tweet originated by a user u will be re-tweeted by her follower v clearly depends on the hashtags and other keywords in that tweet (Figure 1 and [19]). Following an empirical study by Barbieri and Bonchi over the real-world *Last.FM* social network [3], a new song due to collaboration between *Lana del Rey* and *Katy Perry* would reach to more people (by means of information diffusion), than some other song that combines *Metal* and *Electronic* bands. In the context of 2016 US Presidential election, Hillary Clinton's campaign promises were infrastructure rebuild, free trade, open borders, unlimited immigration, equal pay, increasing minimum wage, etc. To get more votes, Hillary's publicity manager could have prioritized the most influential among all these standpoints in speeches, while also planning how to influence more voters from the "blue wall" states (Michigan, Pennsylvania, and Wisconsin) [26]. As speeches should be kept limited due to time constraints and risk of becoming ineffective because of information overload, it is desirable to find a limited set of standpoints that maximize the influence from a set of early adopters (e.g., popular people who are close to Hillary Clinton) to a set of target voters (e.g., citizens of the "blue wall" states) [20].

Motivated by these timely demands, we revisit the classical influence maximization problem, and investigate a novel problem

SIGMOD'18, June 10–15, 2018, Houston, TX, USA

ACM ISBN 978-1-4503-4703-7/18/06…$15.00
https://doi.org/10.1145/3183713.3199670

as follows. *Given a set of target customers, a small budget k on the number of seed users, and a small budget r on the number of relevant tags, what are the top-k seed nodes and the top-r most relevant tags that maximize the expected spread of the campaign within the target set?* Setting a small budget on the number of tags is critical, for example, (a) to avoid information overload against campaign effectiveness (e.g., if one wants to write an advertisement or a blog, she may want to focus on a small number of relevant topics), and (b) for cost-effective planning and due to other physical constraints (e.g., an album can accommodate a *limited* number of genres/ songs).

Case Study. To demonstrate the effectiveness of our problem, we conduct a case study on the real-world *Yelp* social network dataset, downloaded from yelp.com.sg/dataset/challenge. This is a reviewing-based social network, where users can assign reviews to a business (e.g., a restaurant, a travel agency, etc.) after visiting it. Each business is characterized by a few categorical attributes (e.g., Pubs, Seafood, Hotels, Clubs, Arts, etc.), which we use as tags in our problem setting. There are total 195 distinct categories in this dataset. For a pair of friends u and v in the social network, and for their reviews on every common business category, we compute the influence direction and probability — based on their time-stamps of visits and their frequency of reviewing the same category. More details about the dataset can be found in Section 6.

As shown in Table 1, the most relevant tags found by our method, for maximizing the influence spread in each target city, are quite different, e.g., for *Las Vegas*, popular categories include *Dance Clubs*, *Arts & Entertainments*, and *Travel*, whereas for *Pittsburgh*, various food items appear in the top-10 list, including *Specialty Food*, *Seafood*, *Coffee & Tea*, *Ice Cream & Frozen Yogurt*. Hence, if someone wants to promote her hotels via tweets or blog posts in each one of those cities, the best choice of advertisement tags are not the same (entertainment facilities for *Las Vegas*, while foods for *Pittsburgh*). Furthermore, Figure 2 shows that an optimal set of advertisement tags for a specific city may not work in the same way for other cities. It is interesting to note that even only with 10 most relevant tags (found by our algorithm), we can obtain nearly 90% of the influence spread, as compared to that achieved by all 195 tags present in the dataset. This case study demonstrates the usefulness and relevance of our novel query, which we investigate in this paper.

Challenges. Although influence maximization has been recently studied in a topic-aware manner [2–4, 6, 20], our problem of jointly finding the top-k seed users and the top-r relevant tags is a novel one. Learning the parameters of a topic-aware influence diffusion model was discussed in [4, 20]. Chen et. al. [6] and Aslay et. al. [2] studied efficiently finding a set of seed nodes that maximize the spread of information for a given topic set. In contrast, the top-r relevant tags are not known apriori in our problem setting. This is a non-trivial problem, since the **NP**-hardness of the classical influence maximization problem [12] and the #**P**-hardness of the influence spread computation [8] directly follow in our scenario. In addition, even using a polynomial-time oracle for the classical influence maximization problem, with $(1 - 1/e - \epsilon)$ approximation guarantee [5, 28, 29], our problem still remains **NP**-hard, as well as hard to approximate. To this end, we develop an iterative algorithm that alternatively optimizes the top-k seed users and the top-r relevant tags, coupled with smart indexing, optimization techniques, and good initial conditions — targeting both accuracy and efficiency.

Target City	Top-10 Tags
Las Vegas	Arts & Entertainment, Travel, Hotels, Buffets, Dance Clubs, Desserts, Chinese, Mediterranean, Japanese, Burger
Toronto	Canadian, Chinese, Japanese, Pubs, Coffee & Tea, Italian, Arts & Entertainment, Comfort Food, Chiropractors, Physical Therapy
Pittsburgh	Burger, Mexican, Seafood, Grocery, Arts & Entertainment, Italian, Sport Bars, Coffee & Tea, Ice Cream & Frozen Yogurt, Specialty Food

Table 1: Case study: top-10 tags found for influence maximization in three target cities.

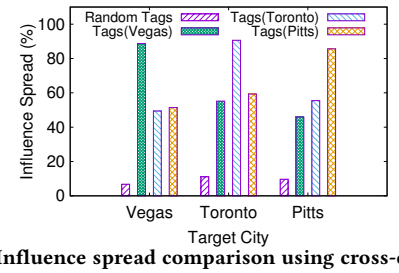


Figure 2: Influence spread comparison using cross-city tags. k=10 seeds, r=10 tags, Y-axis denotes the % of spread by 10 selected tags, over total spread by all 195 tags.

Our Contribution and Roadmap. Our contributions can be summarized as follows:

- We focus, for the first time, on the problem of jointly finding the top-k seed users and the top-r relevant tags that maximize the influence within a given set of target customers in a social network. We show that our problem is **NP**-hard, and also hard to approximate, even when polynomial-time influence estimation and seed set selection methods are employed (Section 2).
- We employ reverse sketching [5, 29] under target influence maximization setting. Moreover, with our smart heuristic indexing method LL-TRS, we reduce the running time by 30% with practically small influence estimation error (Section 3).
- We design batch-based path selection algorithm for the top-r relevant tags finding that significantly improves the accuracy (up to 30%), when compared with existing methods [13] having similar running time (Section 4).
- We combine seeds and tags finding algorithms in an iterative manner, and prove that the influence spread monotonically increases until convergence. Also, we provide smart initialization methods for faster convergence (Section 5).
- We conduct a thorough experimental evaluation with several real-world social networks. It confirms that within a limited number of iterations (3~4), the influence spread by our algorithm converges to a high quality, compared to various baseline methods (Section 6).

2 PRELIMINARIES

2.1 Problem Formulation

In the literature of influence maximization [7, 12], whenever a social network user gets influenced by a campaign (e.g, buys a product, shares a photo, or tweets an information), she is viewed as being activated. A social influence network is modeled as an uncertain graph $\mathcal{G} = (V, E, P)$, where V is a set of n nodes (users), $E \subseteq V \times V$ is a set of m directed edges (friendship links, follower/followee relations, etc.). We denote by C the set of all tags present in the social network. As stated earlier, these could be explicit hashtags, together

with other representative keywords. Appearance of specific tags in an online campaign affects the corresponding influence diffusion probabilities between two users. Specifically, $P : E \times C \rightarrow (0, 1]$ is a function that assigns a conditional probability to every edge $e \in E$ given a specific tag $c \in C$, such that, $P((u, v)|c)$ denotes the probability that an active user u will influence her neighbor v, given the tag c in the campaign (see Figure 1). Such tag-dependent influence probabilities can be learnt from past propagation datasets, e.g., [4, 10, 20]. In this work, we employ the widely used Independent Cascade (IC) model [12] for information diffusion, which we introduce next.

IC Model. In the classical IC model — that is, assuming fixed edge probabilities $P(u, v)$, and without considering different influences due to various tags — the campaign starts with an initially active set of seed nodes, and then unfolds in discrete steps. When some node u first becomes active at step t, it gets a single chance to activate each of its currently inactive out-neighbors v; it succeeds with probability $P(u, v)$. If u succeeds, then v will become active at step $t+1$. Whether or not u succeeds at step t, it cannot make any further attempts in the subsequent rounds. If a node v has incoming edges from multiple newly activated nodes, their attempts are sequenced in an arbitrary order. Also, each node can be activated only once and it stays active until the end. The campaigning process runs until no more activations are possible. Therefore, the IC model assumes the following influence cascading scenario: people get influenced by a campaign when they come in direct contact with their friends who *very recently* adopted that campaign.

It was shown in [12] that the IC model is equivalent to the *possible world* semantics. A possible world $G = (V, E_G)$ is one certain instance of the uncertain graph $\mathcal{G}$, where $E_G \subseteq E$, and is obtained by independent sampling of the edges. Each possible world G is associated with a probability of existence $Pr(G)$ as follows.

$$Pr(G) = \prod_{e \in E_G} P(e) \prod_{e \in E \setminus E_G} (1 - P(e)) \tag{1}$$

Given a set $T \subseteq V$ of customers, the (targeted) influence spread in the possible world G is defined as the number $\sigma_G(S, T)$ of target nodes that are reachable from the seed set S in G, i.e.,

$$\sigma_G(S, T) = \sum_{t \in T} I_G(S, t) \tag{2}$$

In the above equation, $I_G(S, t)$ is an indicator function that takes value 1 if at least one node in S is reachable to t in G, and 0 otherwise. The expected influence spread $\sigma(S, T)$ by the seed set S within the target set T is computed as the expectation of the number of reachable target nodes from S, i.e.,

$$\sigma(S, T) = \sum_{G \sqsubseteq \mathcal{G}} [\sigma_G(S, T) \times Pr(G)] \tag{3}$$

Tag-aware IC Model. In reality, edge probabilities depend on various tags related to the campaign. The information diffusion probability via an edge e, given a set of tags $C_1 \subseteq C$ in the campaign, can be derived as $P(e|C_1) = \mathop{\mathbb{F}}_{c \in C_1}(P(e|c))$. Here, $\mathbb{F}$ is an aggregate function, which can be defined in several ways as follows.

- **Independent Tag Aggregation.** In this case, we assume that the existence of an edge is determined by an independent process (coin flipping), one per tag c, and the ultimate existence of an edge is decided based on the success of at least one of such processes. This assumption is consistent with the IC model [12], and naturally holds if the edge probabilities were learned assuming independence of tags [25]. Formally, the information diffusion probability via an edge e, given a set of tags $C_1 \subseteq C$ in the campaign, can be derived as $P(e|C_1) = 1 - \prod_{c \in C_1}(1 - P(e|c))$.
- **Topic-Based Tag Aggregation.** This model is recently introduced in [20], which follows the earlier topic-aware influence cascade model in [4]. In particular, the model learns a set of hidden topics $Z = \{z_1, z_2, \ldots, z_{|Z|}\}$ from past contents propagated in the social network. For an edge $e = (u, v)$, the topic-aware influence probability $p(e|z)$ denotes how likely u influences v given the topic $z \in Z$ in the campaign. Moreover, the method in [20] learns the probability $p(c|z)$, which is the probability of sampling a tag $c \in C$, given the topic z.

In this work, we shall consider the independent tag aggregation approach, which is also consistent with the IC model. Moreover, in real-world scenarios, it is generally hard to achieve a specific (optimal) topic distribution for a campaign. Instead, we provide the top-k tags directly that a campaigner can associate to maximize the spread of her influence among the target customers. Nevertheless, our hardness results in Section 2.2 also hold for the topic-based tag aggregation method.

Following the IC model, we assume that edge probabilities in the uncertain graph are independent of one another. Hence, given a set C_1 of tags in a campaign, the uncertain graph $\mathcal{G}$ yields 2^m deterministic graphs $G \sqsubseteq \mathcal{G}|C_1$, where each G is a pair (V, E_G), with $E_G \subseteq E$, and its probability of being observed is:

$$Pr(G|C_1) = \prod_{e \in E_G} P(e|C_1) \prod_{e \in E \setminus E_G} (1 - P(e|C_1)) \tag{4}$$

Analogously, the expected influence spread $\sigma(S, T, C_1)$ by the seed set S in $\mathcal{G}$, given the target set T and the tag set C_1, is computed as:

$$\sigma(S, T, C_1) = \sum_{G \sqsubseteq \mathcal{G}|C_1} [\sigma_G(S, T) \times Pr(G|C_1)] \tag{5}$$

Problem Statement. Given the target set T, a budget k on the maximum number of seed nodes, and a budget r on the maximum number of tags, jointly find the top-k seed nodes and the top-r relevant tags such that the influence spread within the target users is maximized.

$$\langle S^*, C_1^* \rangle = \arg\max_{S, C_1} \sigma(S, T, C_1)$$
$$\text{s. t.} \qquad |S| = k, |C_1| = r \tag{6}$$

2.2 Hardness of the Problem

Our problem relies on the classical influence maximization problem, which is **NP**-hard [12] and on the influence spread computation, which is #**P**-hard [8]. As a result, our problem is also hard.

Theorem 1. *Given the target set T and the set C_1 of top-r tags, finding the top-k seed nodes that maximize the influence spread within the target set is* **NP***-hard.*

Theorem 2. *Given the target set T, seed set S, and the set C_1 of top-r tags, finding the influence spread from the seed nodes to the target set is* #**P***-hard.*

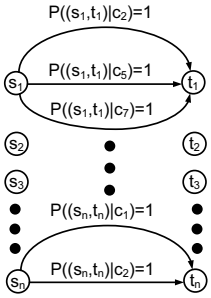


Figure 3: Hardness of top-r tags selection

To prove Theorems 1 and 2, one can construct an updated uncertain graph by only considering the tags in C_1, and thereby assigning fixed edge probabilities.

On the other hand, the classical influence maximization problem (under the IC model) is submodular with respect to inclusion of seed nodes, whereas the influence spread can be estimated in polynomial time via Monte Carlo (MC) sampling (or, a more efficient reverse sketching method [5, 28, 29]), thus a polynomial-time greedy hill-climbing algorithm [12] can solve the classical influence maximization problem with $(1 - 1/e - \epsilon)$ approximation guarantee. Hence, the key question is whether our problem remains hard if a polynomial-time oracle for the classical influence maximization is employed. We show that the answer to this question is positive.

Theorem 3. *Given the target set T and the seed set S, finding the set C_1^* of top-r tags that maximizes the influence spread from the seed nodes to the target nodes is* **NP***-hard, even assuming a polynomial-time oracle for estimating the influence spread.*

Proof. We prove **NP**-hardness by performing a reduction from the **NP**-hard Max r-Cover problem, defined by a collection of subsets $\mathcal{S} = \{S_1, S_2, \ldots, S_m\}$ of a ground set $U = \{u_1, u_2, \ldots, u_n\}$; and the objective is to find a subset $\mathcal{S}^* \subset \mathcal{S}$ of size r such that maximum number of elements in U can be covered by $\mathcal{S}^*$. Now, we construct an instance of our top-r tags finding problem as follows (Figure 3). We create a graph $\mathcal{G}$ and add a set of nodes $t_1, t_2, \ldots, t_n$, one for each element in U. We use them as target nodes. Moreover, we put in $\mathcal{G}$ another set of nodes $s_1, s_2, \ldots, s_n$ (also, one for each element in U), and use them as seed nodes. Finally, if some element $u_i \in U$ is covered by at least one of the subsets in $\mathcal{S}$, we add a directed edge (s_i, t_i) in $\mathcal{G}$. For each set $S_j \in \mathcal{S}$ that covers item u_i, we assign the tag-dependent edge probability $P((s_i, t_i)|c_j) = 1$. Now, we ask if there are r tags by which all target nodes $t_1, t_2, \ldots, t_n$ can be influenced from the seed nodes $s_1, s_2, \ldots, s_n$. Clearly, this corresponds to selecting r subsets from $\mathcal{S}$ that maximize the number of elements covered. Hence, the theorem follows. □

In addition to being **NP**-hard, the problem of finding the top-r relevant tags is also not easy to approximate, since it does not admit any *Polynomial Time Approximation Scheme* (**PTAS**).

Theorem 4. *Given the target set T and the seed set S, the problem of finding the top-r relevant tags that maximize the influence spread from the seed nodes to the target nodes, does not admit any* ***PTAS****, unless* ***P*** *=* ***NP****.*

Proof. See Appendix. □

Moreover, our problem is neither submodular, nor supermodular with respect to inclusion of tags. Thus, a standard greedy hill-climbing algorithm [22] do not directly come with provable approximation guarantees. Non-supermodularity follows from **NP**-hardness (since maximizing supermodular set functions under a cardinality constraint is solvable in polynomial time), whereas we demonstrate non-submodularity with a counter-example.

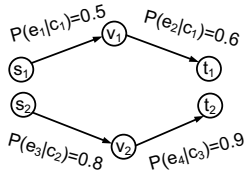


Figure 4: Example of non-submodularity. $P(e|c) = 0$ for all other tag and edge combinations that are not specified.

Lemma 1. *Given the target set T and the seed set S, the problem of finding the top-r tags that maximize the influence spread from the seed nodes to the target nodes, is non-submodular with respect to inclusion of tags.*

A set function f is submodular if $f(A \cup \{x\}) - f(A) \geq f(B \cup \{x\}) - f(B)$, for all sets $A \subseteq B$ and all elements $x \notin B$. Let us consider the example in Figure 4. The seed set = $\{s_1, s_2\}$, and target set = $\{t_1, t_2\}$. Let $C_1 = \{c_1\}$, $C_2 = \{c_1, c_2\}$. We find that $\sigma(S, T, C_1) = 0.5 \times 0.6 = 0.3$, $\sigma(S, T, C_1 \cup \{c_3\}) = 0.3$, $\sigma(S, T, C_2) = 0.3$, and $\sigma(S, T, C_2 \cup \{c_3\}) = (0.5 \times 0.6) + (0.8 \times 0.9) = 1.02$. Therefore, submodularity does not hold in this example.

2.3 Overview of Our Solution

Since our problem of jointly finding the top-k seed nodes and the top-r tags is **NP**-hard, as well as hard to approximate, we design practical solutions aiming for both efficiency and accuracy.

Baseline Greedy Algorithm. This is a baseline approach (given in Section 5): We first select the best seed and the best tag, followed by the second-best seed and the second-best tag, and so on, in a greedy manner. We find that the accuracy of this Greedy method can be improved with an iterative algorithm as follows.

Proposed Iterative Algorithm. We develop an iterative algorithm that alternatively optimizes the top-k seeds (given in Section 3) and the top-r tags (described in Section 4) until it converges to a local optimum. The complete algorithm combining these two steps will be discussed in Section 5. We further consider various initialization criteria for faster convergence. For both top-k seeds and top-r tags selection, we employ *reverse sketching*-based influence maximization techniques [5, 28, 29], which is the state-of-the-art method for achieving good scalability and efficiency.

3 ALGORITHMS FOR FINDING SEEDS

In this section, we design efficient algorithms for finding the top-k seed nodes, assuming that the set C_1 of top-r tags are already provided. Therefore, we derive an updated uncertain graph by only considering the tags in C_1, and thereby assigning fixed edge probabilities. This is identical to the classical (targeted) influence maximization problem; therefore it is **NP**-hard (Theorem 1), but the objective function is monotonic and submodular with respect to the seed set.

Lemma 2. *Given the target set T and the set C_1 of top-r tags, the targeted influence spread (i.e., spread within T) is monotone and submodular with respect to the seed set [15, 27].*

Therefore, one can apply a greedy algorithm (also known as an iterative hill-climbing algorithm) that finds the seed set with an approximation guarantee $(1 - 1/e)$ to the optimal solution [12, 22].

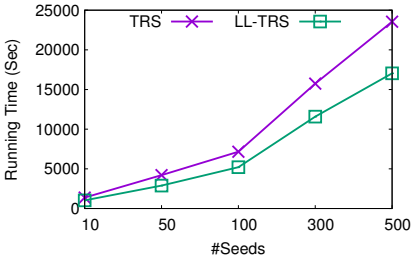


Figure 5: Running time of TRS (state-of-the-art) vs. LL-TRS (our method), *Twitter*, #tags=5, #targets=3K, ϵ=0.1, α=1, δ=0.01, h=3.

At each iteration, the algorithm selects a non-seed node u as a seed node that maximizes the marginal gain. Formally,

$$u = \arg\max_{v \in V \setminus S} [\sigma(S \cup \{v\}, T, C_1) - \sigma(S, T, C_1)] \quad (7)$$

Moreover, we can employ CELF [18] and CELF++ [11] optimizations to further improve the efficiency of the top-k seeds finding.

3.1 Targeted Reverse Sketching

Reverse sketching-based influence maximization techniques, first proposed in [5] and later improved in [28, 29], are the state-of-the-art methods with both good scalability and an $(1 - 1/e - \epsilon)$ approximation guarantee. The algorithms in [5, 28, 29] were designed without any notion of target customers. We first refine it to adjust to targeted influence maximization with the same approximation guarantee $(1 - 1/e - \epsilon)$, and later implement a scalable solution with smart indexing techniques (Sections 3.2 and 3.3).

Let G denote a (deterministic) sub-graph of the input uncertain graph $\mathcal{G}$, generated by removing each edge $e \in E$ with probability $1 - p(e)$ independently. The *Reverse Reachable (RR) Set* for a random target node v consists of all nodes that can reach v in G. The workflow of targeted reverse sketching, which is similar to the original one in [5, 28, 29], is as follows:

1. Generate θ random RR-sets from $\mathcal{G}$.

2. Next, a greedy algorithm repeatedly identifies the node presenting in the majority of RR-sets, adds it to the seed set, and the RR-sets containing it are removed. This process continues until k seed nodes are identified.

In [29], it was proved that when θ is sufficiently large, reverse sketching returns near-optimal results with high probability.

Our major update is that the target node v in each RR-set(G, v) is sampled uniformly at random *only* from the target set T (and not from all nodes). We refer to it as targeted reverse sketching (TRS).

Theorem 5. *If θ satisfies:*

$$\theta \geq (8 + 2\epsilon) \cdot |T| \cdot \frac{\ln n + \ln \binom{n}{k} + \ln 2}{OPT_T \cdot \epsilon^2} \quad (8)$$

then TRS returns a $(1 - 1/e - \epsilon)$-approximate solution, with at least $1 - n^{-1}\binom{n}{k}^{-1}$ probability.

In the above Inequality, OPT_T is the maximum influence spread in the target set T from any node set of cardinality k. Due to the similarity of our Inequality 8 with that of the original Inequality from [29], we omit the proof here (see Appendix for details). However, *our primary contribution for the targeted influence maximization is designing several smart indexing schemes*, whose improvement is demonstrated in Figure 5 and details are introduced next.

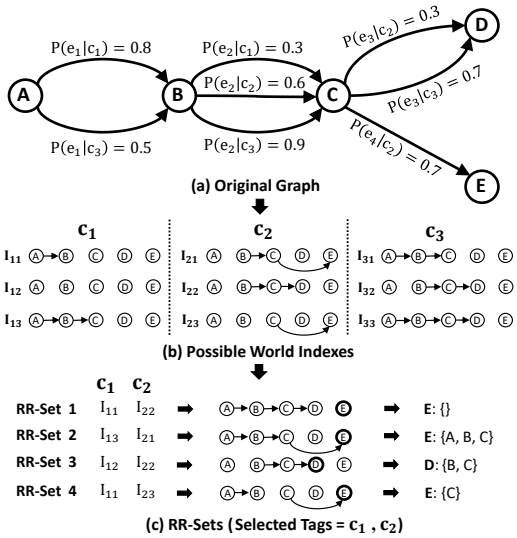


Figure 6: I-TRS indexing and querying: The original uncertain graph contains three tags $\{c_1, c_2, c_3\}$. $P(e|c) = 0$ for all other tags that are not specified. The query considers the target set $T = \{D, E\}$. The selected tag set $C_1 = \{c_1, c_2\}$. We set $\theta = 4$, $\theta_c = 3$.

3.2 Indexing for Targeted Reverse Sketching

Sampling RR-sets online is still time-consuming for larger social networks. Moreover, as stated in Section 2.3, our method works in several rounds — every time finding the seed nodes over a different graph (due to different tag sets), and this requires generating of different RR-sets at each round. Aiming at higher efficiency, we slightly loosen the constraint on RR-set independency (that is required to provide an approximation guarantee for TRS), and instead design heuristic indexing methods that appropriately generate random samples beforehand, thereby significantly reducing the online querying time (up to 30% as shown in Figure 5). With a series of improvements, we develop three indexing schemes, I-TRS, L-TRS, and LL-TRS, and later show that *our ultimate approach LL-TRS supersedes the original RR-set based scheme in online query time, while having a small overhead due to index building and storage.*

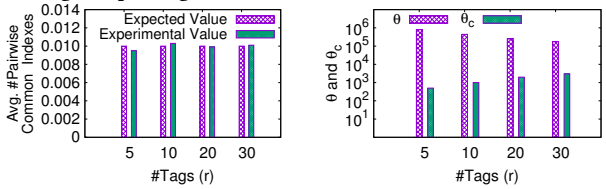
We refer to our first indexing scheme as I-TRS (i.e., Indexing for Targeted Reverse Sketching). Note that in successive iterations of our framework, the optimal tag set may be different. The working graph for an iteration is a combination of each chosen tag's individual uncertain graph. This motivates us to sample and build indexes on uncertain graphs by considering each tag separately. We define our random samples, denoted by *Possible World Index*, as following.

Possible World Index. A possible world index (I, c), for a tag c, is the sub-graph of $\mathcal{G}$, generated by: (a) first keeping only the edges associated with tag c, (b) then removing any remaining edge $e \in E$ with probability $1 - p(e|c)$, and finally (c) retaining all nodes of $\mathcal{G}$ inside G, even if there is no edge connected to some node in G.

Example 1. *Figure 6 demonstrates the workflow of I-TRS on an uncertain graph $\mathcal{G}$ (shown in sub-figure(a)). $\mathcal{G}$ originally contains 3 tags $\{c_1, c_2, c_3\}$. When generating a possible world index for a specific tag c_i, we consider only those edges e in $\mathcal{G}$ such that $p(e|c_i) > 0$; and then remove it with probability $1 - p(e|c_i)$. We record all the edges remained as one possible world index for the tag c_i. Suppose, in this example, we are building $\theta_c = 3$ possible world indexes for every tag c_i. Then, we repeat the aforementioned procedure 3 times*

Method	Inf. Spread in Targets (%), #Seeds(k) = 20				Inf. Spread in Targets (%), #Tags(r) = 20			
	r=5	r=10	r=20	r=30	k=5	k=10	k=20	k=30
TRS	24.70	48.49	71.26	75.11	42.38	61.69	71.48	73.36
I-TRS	24.58	48.6	71.33	74.92	42.51	61.66	71.33	73.29
Deviation	-0.12	+0.11	+0.07	-0.19	+0.13	-0.03	-0.15	-0.07

Table 2: Accuracy comparison between TRS and I-TRS with optimal tags, *Yelp*. #targets=3K, $\delta = 0.01$, $\alpha = 1$.



(a) Avg. #Pairwise Common Indexes (b) θ and θ_c

Figure 7: Implication of Theorem 6, *Yelp*. #targets=3K, #seeds=10, $\delta = 0.01$, $\alpha = 1$. Optimal tags are used.

for each tag. Thus, we obtain all possible world indexes as shown in sub-figure(b). Finally, sub-figure(c) presents how we generate RR-Sets when processing online queries. Assume that one requires $\theta = 4$ RR-Sets and the selected (top-2) tag set is $\{c_1, c_2\}$. Then, we repeat the following steps 4 times: First, randomly select one possible world index from each index set of tag c_1 and c_2, and combine them together as the working graph G. For example, the first working graph in sub-figure(c) is constructed by combination (i.e., graph union) of two possible world index, I_{11} and I_{22}. Next, we choose a random node t from target set T, and apply reverse BFS in G starting from t, recording all visited nodes as an RR-Set. This ultimately produces 4 RR-Sets, $\{\{\}, \{A, B, C\}, \{B, C\}, \{C\}\}$.

Index Size Estimation. Two working graphs will be correlated if they share some common indexes. As an example, in Figure 6(c), the working graphs for RR-sets 1 and 4 are correlated, because they use the same index I_{11} for tag c_1. We, therefore, aim at reducing the average number of common indexes between any pair of working graphs from the set G of all θ working graphs. Note that the following theorem does not ensure approximation guarantee, but only provides a bound on the required number of possible world indexes per tag. It ensures that the average number of common indexes between any pair of working graphs is practically small.

Theorem 6. *Let $C(G)$ denote the random variable corresponding to the average number of common indexes between any two different working graphs selected from the set G of all θ working graphs, built by* I-TRS. *If θ_c satisfies the following, for each tag c,*

$$\theta_c \geq \frac{r\theta}{\alpha\delta(\theta-1)+r} \tag{9}$$

then $C(G) \leq \alpha$ holds with at least $(1-\delta)$ probability.

Proof. $C(G)$ can be calculated as follows.

$$C(G) = \frac{1}{\theta(\theta-1)} \sum_{g_i \in G} \sum_{I_j \in I(g_i)} \sum_{g_t \in G \setminus g_i} \phi(I_j, g_t) \tag{10}$$

where $I(g_i)$ is the set of indexes used by the working graph g_i. The indicator function $\phi(I_j, g_t)$ returns 1 if $I_j \in I(g_t)$ and 0 otherwise.

Let $X(I_j)$ be the random variable denoting the total count of index I_j appearing in all θ working graphs in G. We have:

$$X(I_j) = \sum_{g_i \in G} \phi(I_j, g_i) \tag{11}$$

For each tag, we have θ_c candidate indexes, and they share the same probability to be chosen by a working graph. Therefore, $X(I_j)$ follows the Binomial distribution with probability $\frac{1}{\theta_c}$, that is, $X(I_j) \sim B(\theta, \frac{1}{\theta_c})$ and $\mathbb{E}\left[X(I_j)\right] = \frac{\theta}{\theta_c}$. Then, we can rewrite Equation 10 as:

$$\begin{aligned} C(G) &= \frac{1}{\theta(\theta-1)} \sum_{g_i \in G} \sum_{I_j \in I(g_i)} \left(\sum_{g_t \in G} \phi(I_j, g_t) - \phi(I_j, g_i) \right) \\ &= \frac{1}{\theta(\theta-1)} \sum_{g_i \in G} \sum_{I_j \in I(g_i)} \left(X(I_j) - 1 \right) \\ &= \frac{r}{\theta - 1} \left(X(I_j) - 1 \right) \end{aligned} \tag{12}$$

Taking expectation at both side, we get:

$$\mathbb{E}\left[C(G)\right] = \frac{r}{\theta-1}(\mathbb{E}\left[X(I_j)\right] - 1) = \frac{(\theta-\theta_c)r}{(\theta-1)\theta_c} \tag{13}$$

By Markov's Inequality, we have:

$$\Pr\left[C(G) \leq \alpha\right] \geq 1 - \frac{\mathbb{E}\left[C(G)\right]}{\alpha} \tag{14}$$

Let $\delta = \frac{\mathbb{E}[C(G)]}{\alpha}$, we have:

$$\begin{aligned} \alpha\delta = \mathbb{E}\left[C(G)\right] &= \frac{(\theta-\theta_c)r}{(\theta-1)\theta_c} \\ \implies \theta_c &= \frac{r\theta}{\alpha\delta(\theta-1)+r} \end{aligned} \tag{15}$$

This completes the proof. □

Empirical Study. Table 2 and Figure 7 provide an experimental study for Theorem 6. We show in Table 2 that the difference of influence estimated by TRS (approximation method) and I-TRS (our indexing method) is within 0.2% in all cases. Figure 7(a) provides justification for our good performance. Empirical results confirm that, when we set the probabilistic upper bound $\alpha = 1$ (and $\delta = 0.01$), the average number of pairwise common index is only 0.01, which is similar to its expectation (Equation 13). This means that the average number of common indexes between any pair of working graphs is practically small, which leads to the similar influence estimation as the approximation method, TRS. Moreover, Figure 7(b) shows that our θ_c is much smaller than θ (about $3 \sim 4$ orders of magnitude), leading to smaller index size and building time.

Time Complexity. Following [29], the time complexity of TRS algorithm can be expressed as: $O(k(m+n)(c_1+c_2+pc_3)\log n/\epsilon^2)$, where k is the number of seeds, m and n the number of edges and nodes, respectively. The term $O(k(m+n)\log n/\epsilon^2)$ represents the number of edges examined to ensure $(1-1/e-\epsilon)$ approximation guarantee, whereas $(c_1+c_2+pc_3)$ denotes the cost of processing each edge. In particular, c_1 is the cost to read an edge information (i.e., its endpoints, probability), c_2 the cost of a coin toss (i.e., random number generation) for checking its existence, and c_3 the cost to check whether its source node is visited or not during the reverse BFS, if the coin toss succeeded (with edge probability p). In contrast, for our indexing algorithm, the time complexity is: $O(k(m+n)pt(c_1+c_3)\log n/\epsilon^2)$, where t is the average number of tags per edge. This is because our indexes remove the coin toss cost, instead an edge may exist in multiple tag's indexes. Specifically, for smaller t, we find that $pt(c_1+c_3) < (c_1+c_2+pc_3)$, and this difference is magnified by a large number $O(k(m+n)\log n/\epsilon^2)$ of edges examined. This explains

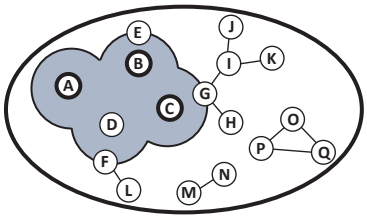


Figure 8: Local indexing: Target set = $\{A, B, C\}$. The shaded region is the local region, on which the possible world indexes are built.

Dataset	Index Size (GB)			Index Building Time (Sec)			Querying Time (Sec)		
	I-TRS	L-TRS	LL-TRS	I-TRS	L-TRS	LL-TRS	I-TRS	L-TRS	LL-TRS
lastFM	3.8	1.41	**1.12**	0.7	**0.3**	**0.3**	24	23	24
Yelp	155.2	14.9	**12.5**	664.1	59.9	**53.1**	2739	2778	2751
DBLP	722.1	72.8	**66.7**	784	98	**78**	885	908	914
Twitter	9893	684	**448**	1724	137	**82**	6446	6577	6494

Table 3: Index size and building time comparison of three indexing methods, #seeds=10 , #tags=10, #targets=3K, $\epsilon = 0.1$, $\delta = 0.01$, $h = 3$.

the efficiency improvement due to our indexing method over the state-of-the art TRS algorithm by a margin of 30% (Figure 5).

3.3 Lazy and Local Indexing: For Improved Efficiency and Storage

Lazy Indexing. Our previous indexing method I-TRS builds θ_c indexes for every tag in the original graph in advance. In practice, only a few tags are actually involved during query processing. Taking our experiments on *Yelp* dataset as an example, if $r = 10$, our algorithm terminates after 3 rounds, there will be at most 30 tags, and only their indexes are involved in the entire query processing. However, in total there are 195 tags in *Yelp*; therefore, I-TRS builds at least 165 sets of useless indexes here. Motivated by such fact, we propose the *lazy* indexing strategy, denoted as L-TRS, as follows.

Assume that for the current iteration of our framework, the optimal tag set is given by C_1. For some tag $c \in C_1$, if possible world indexes for tag c already exist from earlier rounds, we also use them as indexes in the current round. Only for other tags $c \in C_1$ for which an index set does not exist, we build θ_c possible world indexes. In other words, we create indexes for a tag in a lazy manner as and when they are necessary. Therefore, *L-TRS improves on I-TRS in both indexing time and space. Another aspect of our lazy indexing is that it makes indexing useful even for one-time querying*, by generating only the necessary sets of indexes, while permitting re-using of indexes across different iterations of our framework.

For lazy indexing to be effective, we prove the following claim.

Lemma 3. *If a tag c was encountered in previous iterations, there is no need to build more indexes for c in the current round.*

Proof. See Appendix. □

Local Indexing. In many real-world application scenarios, the target nodes tend to be clustered locally. For example, in the *Yelp* dataset, "users from a specific city" is a common way to define a target set. The users from *Toronto* are located closely in some region of the graph, while the users from *Pittsburgh* are in another cluster. When doing the reverse BFS in a (deterministic) graph starting from a *Toronto* target user, most of the nodes visited generally also come from the *Toronto* cluster, and we rarely encounter a *Pittsburgh* user in the corresponding RR-set. Inspired by this observation, we further implement a *local* indexing strategy as follows.

Given an uncertain graph $\mathcal{G}$, a target set T, and a small distance threshold h, we define the *local region* as the subgraph $\mathcal{G}_l$ of $\mathcal{G}$, where all nodes in $\mathcal{G}_l$ are at most h-hops away from at least one target node in T. The subgraph $\mathcal{G} \setminus \mathcal{G}_l$ is denoted as the *outer region*. During the reverse BFS traversal and RR-sets computation, since the nodes within the local region are visited predominantly compared to the nodes outside, we only index the local region of the uncertain graph, that is, we build our possible world indexes only considering the local region. We note that h is a user-defined distance threshold, and we select its optimal value (e.g., h=3) based on empirical results.

Example 2. *In Figure 8, the shaded area around the target set $\{A, B, C\}$ is the local region. Nodes $\{A, B, C, D, E, F, G\}$ are in the local region, and E, F, G are boundary nodes. The remaining nodes $\{H, I, J, ..., Q\}$ are outside the local region (i.e., in the outer region). We only build indexes over the local region, which is much smaller than the indexes over the entire graph. Through boundary nodes in the local region, the online query processing is still able to visit outside nodes, e.g., I, J, K, which may be included in a limited number RR-sets. Clearly, these outside nodes are less frequently visited, hence we do not build indexes over them. Moreover, there are several nodes, e.g., M, N, O, P, Q, that are disconnected from the target set in the input uncertain graph; hence, they can never be included in any RR-set. By constructing indexes locally, we avoid the cost of redundant index building over such disconnected nodes.*

The ultimate indexing scheme that we propose in this work is referred to as the LL-TRS (LL is an abbreviation for *lazy and local*).

Benefits of LL-TRS Index. We note that *LL-TRS ensures the same guarantee in terms of the average number of common indexes between any pair of working graphs, as in Theorem 6.* On the other hand, LL-TRS index is query-specific, since one needs to know the target set to compute the local region, and build the index only over that local region (also in a lazy manner). Therefore, for fairness of comparison, we add the index building time, while reporting the query answering time for both L-TRS and LL-TRS approaches. By well choosing h, our all three index based approaches require similar querying time (Table 3). Table 3 also compares the index size and indexing time among our indexing methods. We find that LL-TRS significantly reduces indexing time and space.

4 ALGORITHMS FOR FINDING TAGS

Given a seed set S, a target set T, and a budget r on the number of tags, we develop algorithms to find the top-r tags that maximize the expected influence spread from S to T. The problem is **NP**-hard, hard to approximate, and neither submodular, nor supermodular for inclusion of tags (Section 2.2). Thus, unlike our top-k seeds finding algorithm in the earlier section, a similar hill-climbing approach does not come with a provable approximation guarantee. Therefore, *we adopt a greedy heuristic analogous to [13], and improve its quality (by a margin of 30%) with a novel batch-paths selection strategy.*

We briefly introduce the individual paths selection method [13] in Section 4.1, point out its shortcomings in Section 4.2, and finally describe our novel batch-paths selection algorithm in Section 4.3.

4.1 Individual Paths Selection

Intuitively, what matters in computing the influence spread from the seed set S to the target set T is the set of highly probable paths connecting them [8, 15, 16], where the probability of a path is determined by multiplying the edge probabilities on that path. Motivated by this, a two-step approach could be developed as follows.

- First, we select the top-l most probable paths between every seed-target pair (in a tag agnostic manner). It could be

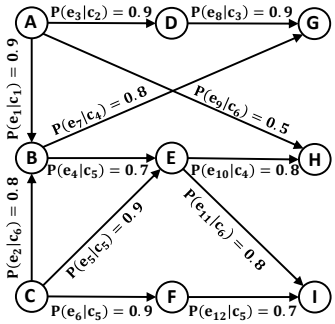


Figure 9: Tags selection. Seed set = $\{A, B, C\}$, target set =$\{G, H, I\}$.

implemented with the Eppstein's algorithm [9, 13]. Let us denote by $\mathcal{P}$ the set of all paths selected as above.

- Next, we iteratively include these paths from $\mathcal{P}$ into our solution $\mathcal{P}_1$ that maximally increase the influence spread (estimated via MC-sampling) from S to T, while still maintaining the budget on total r tags over all included paths.

Since the algorithm includes only one path at a time in the second step, we refer to this method as *individual paths selection.*

4.2 Shortcomings of Individual Paths Selection

We demonstrate with the following example that the individual paths selection approach has several critical shortcomings.

Example 3. *In Figure 9, there are total 14 paths between the seed set $\{A, B, C\}$ and the target set $\{G, H, I\}$. They are as follows.* $A \to G : \{e_3e_8, e_1e_7\}$, $A \to H : \{e_1e_4e_{10}, e_9\}$, $A \to I : \{e_1e_4e_{11}\}$, $B \to G : \{e_7\}$, $B \to H : \{e_4e_{10}\}$, $B \to I : \{e_4e_{11}\}$, $C \to G : \{e_2e_7\}$, $C \to H : \{e_2e_4e_{10}, e_5e_{10}\}$, $C \to I : \{e_2e_4e_{11}, e_5e_{11}, e_6e_{12}\}$. *Let us apply the individual paths selection algorithm to find the top-3 tags in this example. In the first iteration, path e_3e_8 will be included in the solution, since it has the highest individual expected spread $\sigma(S, T, \{e_3e_8\}) = 0.81$. In the second round, because of budget on tags: $r = 3$, there are only three choices: $\{e_7, e_9, e_6e_{12}\}$. Considering the marginal gain, e_6e_{12} will be included. Therefore, we obtain the top-3 tag set $\{c_2, c_3, c_5\}$ The expected spread achieved by this top-3 tag set is: $\sigma(S, T, \{e_3e_8, e_6e_{11}\}) = 0.81 + 0.63 = 1.44$.*

Unfortunately, the best 3 tags in this example are c_4, c_5, c_6, which result in an expected influence spread $\sigma(S, T, \mathcal{P}_1) \approx 2.61$, where $\mathcal{P}_1 = \{e_7, e_9, e_2e_7, e_4e_{10}, e_5e_{10}, e_2e_4e_{10}, e_5e_{11}, e_2e_4e_{11}, e_6e_{12}\}$. It is obvious that in Example 3, the individual paths inclusion algorithm makes a bad choice in the first round, and can never recover from it. Below we investigate the key reasons behind this problem.

(1) Although the path e_3e_8 has the highest probability (0.81) among all paths, the tags on this path, c_2 and c_3, do not appear elsewhere in the graph. Our final goal is to select optimal tags, but not the paths themselves. As an example, the path e_4e_{10} has only 0.56 probability; however, selecting e_4e_{10} will also activate paths e_7, e_5e_{10}, and e_6e_{12}, since they all share same tags c_4 and c_5, leading to a total expected spread $\sigma(S, T, \{e_4e_{10}, e_5e_{10}, e_6e_{12}, e_7\})$ $=0.8[1-(1-0.7)(1-0.9)]+0.9\times0.7+0.8 \approx 2.21$. This implies that paths sharing same tags must be evaluated together.

(2) Though the path e_7 has a slightly lower probability (0.8) compared to that of e_3e_8 (0.81), including it only costs 1 tag, while e_3e_8 consumes 2 tags. Therefore, it is more appropriate to evaluate the marginal gain per new tag included, rather than that of a path.

(3) Some paths, e.g., e_7, e_1e_7, and e_2e_7, are actually equivalent in the original graph. All the source nodes shall be activated in the beginning, therefore the existence of edges between them has no impact on influence diffusion. By removing edges between source nodes beforehand, we can avoid including redundant paths, which can significantly reduce the computation cost (only 8 out of 14 paths remaining for Example 3).

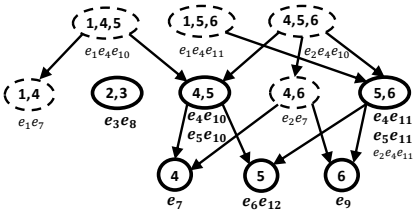


Figure 10: Batch-paths lattice for Figure 9. Dashed circles present redundant search space due to edges between seeds (removed).

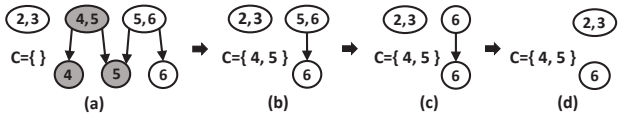


Figure 11: Updates of batch-paths lattice on inclusion of tags $\{4, 5\}$

Inspired by above, we develop a batch-paths selection algorithm that achieves higher accuracy with the included tags.

4.3 Batch-Paths Selection

We start with defining a *path-batch* $\mathcal{P}(C)$ for a tag set C, which is a set of paths such that for every path $P \in \mathcal{P}(C)$, its tag set is equal to C. For example, $\mathcal{P}(c_4, c_5) = \{e_4e_{10}, e_5e_{10}\}$, whereas $\mathcal{P}(c_5) = \{e_6e_{12}\}$. Clearly, path-batch is our measuring unit in the novel algorithm. All paths located in the same path-batch contain same tags. Hence, activating one of them will activate other paths in the batch. Furthermore, since the tag sets of path-batches can have subset relationships, we should also find those batches which are "dominated" by the current batch (based on subset relationships over tag sets), and activate them. For example, activating $\mathcal{P}(c_4, c_5)$ will activate all paths in the path-batch $\mathcal{P}(c_5)$. We define the *descendent* of a path-batch $\mathcal{P}$ as:

$$Des\mathcal{P}(C) = \{\mathcal{P}(C_1) : C_1 \subseteq C\} \tag{16}$$

We are now ready to define the objective of our greedy approach, that iteratively includes a path-batch $\mathcal{P}^*$ having the maximum marginal gain ratio (i.e., marginal gain/new tag), while still maintaining the budget r on the number of included tags.

$$\mathcal{P}^* = \underset{\mathcal{P} \in \mathcal{PB} \setminus \mathcal{PB}'}{\arg\max} \left[\frac{\sigma(S, T, Des\mathcal{P} \cup \mathcal{PB}') - \sigma(S, T, \mathcal{PB}')}{C(\mathcal{P})} \right] \tag{17}$$

In the above equation, $\mathcal{PB}$ is the collection of all path-batches, whereas $\mathcal{PB}'$ is the set of already included path-batches in our solution. $Des\mathcal{P}$ denotes the descendent of the path-batch $\mathcal{P}$, and $C(\mathcal{P})$ is the number of new tags in the path-batch $\mathcal{P}$.

Efficient Algorithm. To find the optimal path-batch at every iteration (by following Equation 17), we build a *path-batch lattice* as shown in Figure 10. (1) All the edges between source nodes are removed beforehand. (2) Each node in the lattice represents a path-batch, labeled by its tag set. (3) In each level of the lattice, the nodes (i.e., path-batches) share same number of tags. (4) Each node in the lattice points to all the nodes in the successive lower level whose tag set is a subset of its own.

To construct the lattice, we consider the paths in $\mathcal{P}$, and filter out those paths which violate the budget r on the number of tags. The

Algorithm 1 Batch-Paths Selection Algorithm

Require: path set $\mathcal{P}$ from source set S to target set T, a positive integer r
Ensure: A subset of paths $\mathcal{P}_1 \subseteq \mathcal{P}$ and its tag set C_1 whose size is not larger than r

```
P_1 = φ, C_1 = φ
Construct path-batches PB and the lattice with P
while |PB| > 0 and C_1 < r do
    Find optimal path-batch P* to include next (Equation 17)
    Add all the paths in P* and its descendants into P_1
    Add tags of P* to C_1
    Update the lattice to remove tags of P* from it (Figure 11)
end while
return Selected tag set C_1
```

lattice is initially empty. For each path in $\mathcal{P}$, if its tag set is the same as some existing batch in the lattice, we add the path into that batch. Otherwise, we create a new batch containing this path. When a new path-batch is generated, we locate its appropriate level within the lattice according to its tag set size. Finally, we add links from each node to all other nodes in the successive lower level whose tag set is a subset of its own. The complete procedure of finding the top-r tags with batch-paths inclusion is given in Algorithm 1. Below we demonstrate it with an example which iteratively includes path-batches from the lattice into our solution.

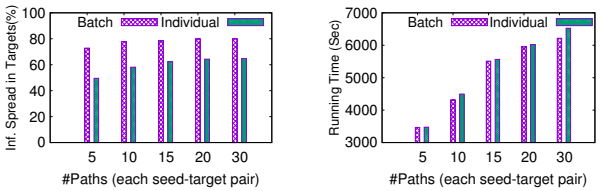
Example 4. *Let us apply the batch-paths inclusion technique to find the top-3 tags in Figure 9. The corresponding lattice is presented in Figure 10. We select the path-batch (with its descendent) $Des\mathcal{P}(c_4, c_5) = \{e_4e_{10}, e_5e_{10}, e_7, e_6e_{12}\}$ as the best batch in the first round (by following Equation 17), which generates an expected spread $\sigma(S, T, Des\mathcal{P}(c_4, c_5)) = 0.8[1 - (1 - 0.7)(1 - 0.9)] + 0.9 \times 0.7 + 0.8 \approx 2.21$. After the inclusion of $Des\mathcal{P}(c_4, c_5)$ in our solution, we update the lattice with the remaining tags and get new one as shown in Figure 11. Since only one more tag can be selected (as r=3), the batch $\{c_6\}$ is selected by following Equation 17). The final tag set selected by our batch-paths inclusion method is $\{c_4, c_5, c_6\}$, and they can bring 2.61 total expected spread, which is higher than that of the individual-paths inclusion (1.44).*

4.4 Indexing for Influence Spread Computation

With increasing number of paths in the batch-paths selection algorithm, MC sampling becomes expensive. Therefore, we apply a reverse sketching-based indexing method to speed up the influence spread computation of path-batches. The major challenge here is that OPT_T would be too small when few paths are selected (at the initial few rounds), which will result in too big θ, i.e., the number of RR-sets required (see Inequality 8). In contrast, MC sampling runs faster when only a few path-batches have been included. Hence, we design a two-step influence spread computation strategy as follows.

(1) When influence spread is less than a given threshold OPT'_T, we adopt MC sampling to compute influence spread of the included path-batches. (2) After the influence spread of included path-batches (estimated by MC sampling) reaches OPT'_T, we switch to reverse sketching for influence spread computation.

The threshold OPT'_T is decided empirically, and can be defined as a ratio over the target set size $|T|$. Therefore, one can compute θ by Inequality 8 without knowing the input target set T apriori. With the inclusion of new path-batches, OPT_T can only increase, which means that θ number of RR-set indexes generated considering the threshold OPT'_T, would be sufficient enough for the successive rounds of the batch-paths selection algorithm. Finally, θ RR-set indexes are created in an offline manner.



(a) Influence spread achieved (b) Running time

Figure 12: Accuracy and efficiency of Batch-paths vs. individual paths selection: *Twitter*, #seeds=10, #tags=10, ϵ=0.1, #targets=3K.

Time Complexity. We denote the set of paths as $\mathcal{P}$ and the collection of path-batches as $\mathcal{PB}$. The time complexity to construct the lattice is $O(|\mathcal{P}| \cdot |\mathcal{PB}| + |\mathcal{PB}|^2)$, the first term is due to grouping the paths into batches and forming the nodes of the lattice, and the second term is due to adding links in the lattice.

The procedure to find the top-r tags by following Algorithm 1 requires $O(r|\mathcal{PB}|w(n'+m'))$ time when using MC-sampling, where w is the number of MC samples, n' and m' are the number of nodes and edges, respectively, of the subgraph induced by the paths in $\mathcal{P}$. The term $w(n' + m')$ is due to estimating the influence spreads of different path-batches via MC-sampling. If using indexed reverse sketching method, it requires $O(\theta|\mathcal{P}| + r|\mathcal{PB}||\mathcal{P}|)$. The first term is for checking each path's existence in each RR index. The second term is due to computing each batch's coverage in the RR indexes. We note that the number of paths in each batch is at most $|\mathcal{P}|$. Therefore, the total complexity is $O(\theta|\mathcal{P}|+a|\mathcal{PB}||\mathcal{P}|+b|\mathcal{PB}|w(n'+m'))$, where $a + b = r$, the budget on the number of tags.

4.5 Benefits of Batch-Paths Selection

Figure 12 shows the accuracy and running time comparison on *Twitter* dataset between batch-paths vs. individual paths algorithms. We find that path inclusion in batches significantly outperforms the individual paths inclusion method in accuracy (up to 30% more influence spread) with comparable running time.

5 PUTTING EVERYTHING TOGETHER

5.1 Baseline Greedy Algorithm

A straightforward greedy algorithm would be to first find the best seed assuming all tags. Then, retrieve the best tag between this seed and all targets. Later, according to the best seed and the best tag selected earlier, we find the next-best seed and the next-bast tag, repeating this procedure until k seeds and r tags are found. We empirically find that this greedy approach results in lower influence spread, since the tags and seeds are selected incrementally — all seeds and tags are not optimized at the same time.

5.2 Proposed Iterative Algorithm

The workflow of our proposed iterative framework is given in Algorithm 2. We first initialize the seed set and the tag set (as discussed in Section 5.3). Next, the framework alternatively and in an iterative manner, optimizes the top-k seed users and the top-r relevant tags, assuming that the others are fixed. These are computed by the seeds finding algorithm and the tags finding algorithm, respectively. At the end of every round, we verify if a fixed-point is reached, i.e., the top-k seeds and the top-r tags achieve a similar influence spread in two successive rounds. Our proposed iterative algorithm,

Algorithm 2 Complete Algorithm for Jointly Finding k-Seeds and r-Tags

Require: Graph $\mathcal{G} = (V, E, P)$, target set T, budgets k and r
Ensure: Optimal seed set S^* and tag set C^*
1: Initialize S^* of size k, C^* of size r
2: **while** not converge **do**
3: Find optimal S^* assuming given C^*
4: Find optimal C^* assuming given S^*
5: **end while**
6: **return** S^*, C^*

in essence, is similar to various classical data mining and machine learning algorithms, e,g., K-means and Expectation-maximization. We prove that our method converges to a local optimum, because at every round, the expected influence spread within the target set monotonically increases until convergence.

Theorem 7. *During the course of our iterative algorithm, the expected spread within the target set monotonically increases.*

Proof. Let $S^{(t)}, C_1^{(t)}$ denote the sets of top-k seeds and the top-r tags, respectively, at the start of the t-th iteration. In the current iteration, we first optimize the seed set, given the tag set. Therefore,

$$\sigma(S^{(t+1)}, T, C_1^{(t)}) \geq \sigma(S^{(t)}, T, C_1^{(t)}) \tag{18}$$

Next, we optimize the tag set, given the seed set as obtained above. Hence, the following holds.

$$\sigma(S^{(t+1)}, T, C_1^{(t+1)}) \geq \sigma(S^{(t+1)}, T, C_1^{(t)}) \tag{19}$$

Combining Equations 18 and 19, the expected spread within target set monotonically increases in successive iterations. □

Finally, we note that our algorithm iterates the objective function $\sigma(S, T, C_1)$ that is bounded above, and whose domain is a finite set. Coupled with Theorem 7, we guarantee that the algorithm converges to a local optimum after a finite number of iterations.

5.3 Initialization for Iterative Algorithm

Uniform at Random Initialization. A simple way to initialize the seed set and the tag set is uniform at random selection from the set of all nodes and all tags, respectively. Intuitively, this initializing method is cheap, but might not always be of good quality, leading to higher convergence time for Algorithm 2. Below, we propose the following initialization methods aiming for faster convergence.

Influence Maximization-based Seeds Initialization. We run a classical influence maximization algorithm (e.g., [29]) to find the top-k seed nodes that maximize the influence spread over the target set, while considering all tags. This leads to a good-quality initial seed set (and therefore, faster convergence); however, the initialization phase becomes more expensive.

Frequency-based Tags Initialization. Generally, those tags appearing frequently among target nodes' incident edges are more relevant. Therefore, we aggregate the total probability for each tag to appear in all target nodes' incident edges. Next, we select the top-r tags with the highest accumulative probability as initial tags.

In our experiments, we compare some combination of the aforementioned initializing methods, and analyze their trade-offs.

Frequency-based Search Space Elimination. We compute the aggregated probability for each tag. Tags with low aggregated probability shall be either in too few edges, or having too low edge probabilities. Such tags generally contribute less in influence diffusion, and we remove them beforehand from the search space.

Name	#Nodes	#Edges	#Tags	Edge Prob: Mean, SD, Quartiles
lastFM	1 322	13 842	78	0.26, 0.23, {0.06, 0.2, 0.41}
DBLP	704 266	4 727 290	230	0.26, 0.15, {0.18, 0.18, 0.33}
Yelp	125 368	808 909	195	0.33, 0.25, {0.18, 0.26, 0.5}
Twitter	6 294 565	11 063 034	500	0.27, 0.14,{0.18, 0.18, 0.33}

Table 4: Characteristics of datasets.

6 EXPERIMENTAL RESULTS

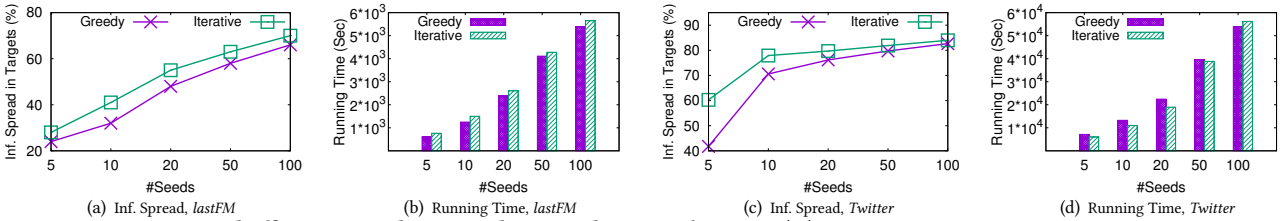
6.1 Experimental Setup

We perform experiments to demonstrate the accuracy (i.e., influence spread within target set), efficiency, and scalability of our algorithms (averaged over 10 runs). The code is implemented in C++ and executed on a single core of a 256GB, 2.40GHz Xeon server.

Dataset. We use four real-world social networks (Table 4). **(1) lastFM.** We obtain this dataset from [4], which is a music-listening history record. Nodes denote users and edges represent their friendships. **(2) DBLP.** *DBLP* (http://dblp.uni-trier.de/xml) is a well-known collaboration network. We downloaded it on March 31, 2017. Each node is an author and edges denote their co-author relations. **(3) Yelp.** *Yelp* (https://www.yelp.com/dataset_challenge) is a user-business reviewing network. Every node is a user. Edges are generated based on their friendship, and directed according to the time-stamps of their reviews for same businesses. **(4) Twitter.** In *Twitter* (http://snap.stanford.edu/data/), nodes are users and edges are their re-tweet relationships. *lastFM* and *DBLP* graphs are undirected, whereas *Yelp* and *Twitter* are directed.

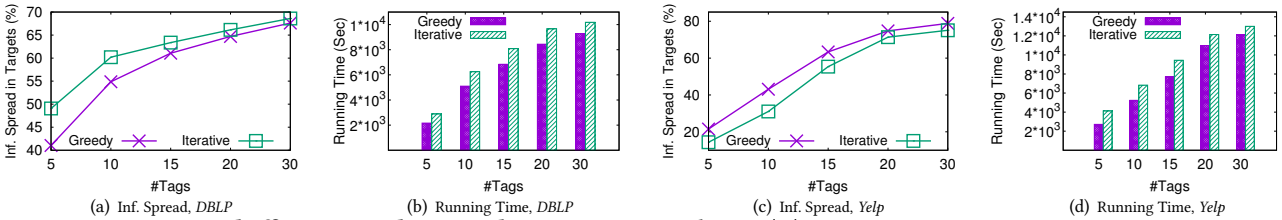
Tags and Edge Probabilities. (1) lastFM. The tags are given by music styles. **(2) DBLP.** The tags are extracted from all paper titles, e.g., database system, neural network, FPGA, etc, based on both their frequency and how well they can represent various sub-areas of computer science. **(3) Yelp.** Users may visit same businesses and each business belongs to some categories (e.g., travel, dance clubs, burger, etc.). We use these categories as tags. **(4) Twitter.** Tags are given by hashtags. For every edge (u, v) and tag c, we compute the frequency (i.e., number of occurrences) of c in (u, v). Let us denote this frequency by t, then we assign probability $p((u, v)|c) = 1-e^{-t/a}$ [23]. The intuition is that the more the times the tag c appears on the edge (u, v), the higher the chance (i.e., the probability) that u influences v for that tag. As an example, for *lastFM* we calculate the frequency t for every music style c based on each node pair's common listening history. We set $a = 5$ for *DBLP* and *Twitter*, $a = 10$ for *Yelp*, and $a = 1000$ for *lastFM* (since each node pair has a large common listening history, we set higher a in this dataset to avoid unusually high connectivity between node pairs).

Target Set. We do breath first search (BFS) from nodes with high in-degrees, and record the visited nodes as targets (since these nodes are co-located within a small region of the graph). **(1) lastFM & Twitter.** There is no specific attribute (e.g., geolocation) for users in these datasets. Hence, we do BFS starting from high in-degree nodes. The target set size of *lastFM* is 500 and that of *Twitter* is varied from 1K to 50K. **(2) DBLP.** We select active databases/ data mining researchers as target nodes, which is decided by whether an author has more than 5 papers in [SIGMOD, VLDB, KDD, PODS, ICDE, CIKM]. Roughly, there are 3K such authors. **(3) Yelp.** We



(a) Inf. Spread, *lastFM* (b) Running Time, *lastFM* (c) Inf. Spread, *Twitter* (d) Running Time, *Twitter*

Figure 13: Accuracy and efficiency analysis to adjust seed set size k. $r = 10$, $|T| = 3K$.



(a) Inf. Spread, *DBLP* (b) Running Time, *DBLP* (c) Inf. Spread, *Yelp* (d) Running Time, *Yelp*

Figure 14: Accuracy and efficiency analysis to adjust tag set size r. $k = 20$, $|T| = 3K$.

Method	Inf. Spread in Targets (%) # Seeds (k)					Running Time (Sec) # Seeds (k)				
	5	10	20	50	100	5	10	20	50	100
RS+RT	42	64	75	**82**	**84**	6 400	11 100	17 100	27 200	39 400
IMS+RT	42	66	75	**82**	**84**	5 800	9 500	14 800	23 500	34 200
RS+FT	48	**74**	**78**	**82**	**84**	**3 700**	**7 700**	**12 000**	**20 100**	**30 200**
IMS+FT	**49**	**74**	**78**	**82**	**84**	5 000	9 200	14 000	22 300	33 100

Table 5: Accuracy and efficiency for different initialization methods, *Yelp*. $r = 20$, $|T| = 3K$.

Method	Inf. Spread in Targets (%) # Iterations							
	1	1.5	2	2.5	3	3.5	4	4.5
RS+RT	17	42	53	60	63	**64**	**64**	**64**
IMS+RT	42	56	62	**66**	**66**	**66**	converged	converged
RS+FT	60	68	**74**	**74**	**74**	converged	converged	converged
IMS+FT	64	71	**74**	**74**	**74**	converged	converged	converged

Table 6: Influence spread achieved after various iterations, with different initialization methods, *Yelp*. $k = 10$, $r = 20$, $|T| = 3K$.

adopt aforementioned BFS method to select active users in each city. Group sizes vary from 1K to 10K.

Parameters Setup. (1) Seed set size (k). We vary k from 5 to 100. **(2) Tag set size (r).** We vary r from 5 to 30. **(3) Target set size ($|T|$).** By default, as the target set, we use 500 nodes for *lastFM*, and 3K nodes for other datasets. We, however, demonstrate scalability by considering up to 50K target nodes. **(4) RR-Sketches.** ϵ is set as 0.1 following [29]. Probability parameter δ and upper bound of common index parameter α in Theorem 6 are set as 0.01 and 1, respectively, and the local-region parameter h as 3. We also verify the sensitivity of our methods with respect to these parameters in the Appendix (Section D.1). **(5) Tags finding.** Number of paths per seed-target pair for our Batch-Paths selection algorithm is set as 10, the optimality of which was demonstrated earlier in Figure 12.

Competing Methods and Initialization. We compare our baseline greedy method with the proposed iterative algorithm. For the iterative algorithm, we consider four initialization techniques. We denote random initialization on seeds and tags as *RS* and *RT*, respectively. Influence maximization-based seed initialization is denoted as *IMS*, and frequency-based tags initialization is denoted as *FT*. We find that the best initializing method is *RS* + *FT*, while other combinations will be evaluated in Section 6.2.

6.2 Accuracy and Efficiency Analysis

We present influence spread within target set (accuracy) and running time (efficiency) comparison between baseline greedy and our iterative algorithms, while varying two major parameters, seed set size k and tag set size r, under different initializing conditions.

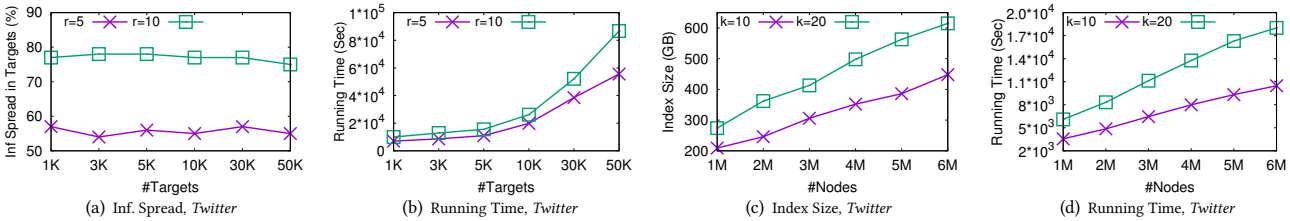
Varying Seed Set Size k. As shown in Figure 13, *the influence spread of our method significantly outperforms the baseline in accuracy, and executes in similar time.* Influence spread rises with larger k. Such growth is more significant when k is small for larger datasets, for example on *Twitter* between $k = 5$ and $k = 10$ the influence increase is from 62% to 77%. On the contrary, the running time is more sensitive to larger k, whose increasing tendency is near linear to the growth of k. For seeds finding part, Theorem 5 requires bigger θ for bigger k (generally less than twice when k doubles). For tags finding part, $2k$ seed size induces, on average, twice the number of paths, which roughly requires double evaluating time. Therefore, our running time increases linearly with bigger k.

Varying Tag Set Size r. For accuracy, the increasing tendency of r is similar to that of k. *With a few important tags, we can already influence most targets. As an example, only top-20 tags can influence about 70% targets in the* Yelp *dataset (Figure 14(c)). Moreover, the influence spread of our iterative algorithm significantly outperforms the greedy baseline in influence spread.*

The running time increases at a higher rate when r is small, which is because the change in influence spread is significant. When r is large enough, adding more tags changes little in the influence spread, leading to a lower increase rate in the running time.

Varying Initialization. As shown in Table 5, different initializing methods eventually lead to similar influence spread. In some specific cases, e.g., with $k = 5$ or 10 seed nodes in *Yelp*, starting with *RT* gets stuck in a local optimum, which is about 8% less than initializing with *FT*. However, with large enough k (e.g., $k = 50$ or 100), high-quality seeds and tags are always found in our solution.

When considering efficiency, initializing with uniform random seeds and tags (RS+RT in Table 6) usually requires 1 round to initialize, about 2.5 rounds to achieve local optimal case, and 1 more round to confirm the convergence. Meanwhile, we observe that either initializing seed set with *IMS* or tag set with *FT* will lead to 1-1.5 round reduction in finding the local optimal solution for most cases (i.e., in total 3 rounds as shown in Table 6). But initializing with *IMS* does not reduce the running time significantly, since applying the classical influence maximization algorithm on the entire graph is expensive. On the other hand, *FT*'s time cost is trivial, and it reduces the running time by minimizing the graph size with already reasonable-quality initial tag set. Therefore, *RS* + *FT* is the best



(a) Inf. Spread, *Twitter* (b) Running Time, *Twitter* (c) Index Size, *Twitter* (d) Running Time, *Twitter*

Figure 15: Scalability analysis. $k = 10$, $r = 10$.

initialization method in terms of both efficiency and accuracy, and we employ this as the default setting in other experiments.

6.3 Scalability Analysis

We test the scalability of our algorithm in two aspects — with larger graph size and with larger target sets. We conduct both these experiments on our largest *Twitter* dataset.

Target Set Size $|T|$**.** The target set size on *Twitter* is varied as 1K, 3K, 5K, 10K, 30K, and 50K nodes. Our results in Figure 15(a) show that *the influence spread percentage within the target set remains consistent, which demonstrates the robustness of our algorithm with respect to different number of target users. The running time in Figure 15(b) increases almost linearly with larger target set size* $|T|$.

Graph Size. The *Twitter* dataset has about 6.3 million nodes, we select 1M, 2M, 3M, 4M, 5M, and 6M nodes uniformly at random to generate 6 graphs, and apply our algorithm on them. Figure 15(c) and Figure 15(d) demonstrate that *both index size and querying time increase linearly with the number of nodes, which confirms good scalability of our algorithm.*

7 RELATED WORK

We categorize related work as follows.

Influence Maximization. Kempe et al. [12] addressed the problem of influence maximization in a social network as a discrete optimization problem, which is to identify the set of seed nodes, having cardinality k, that maximizes the expected influence spread in the graph. Due to **NP**-Hardness of the problem, they proposed a hill climbing greedy algorithm, with an accuracy guarantee of $(1-1/e)$. They also used MC-simulation for estimating the influence spread from a seed set. However, later it was proved in [8] that the exact computation of influence spread is #**P**-Hard.

After that, many algorithms (see [7] for details) have been developed, both heuristic and approximated, to improve the efficiency of the original greedy method. Leskovec et al. [18] and Goyal et al. [11] exploited the sub-modularity property of the greedy algorithm, and proposed more efficient CELF and CELF++ algorithms, respectively. Chen et al. [8] avoided MC simulations, and developed the maximum influence arborescence (MIA) model using maximum probable paths for the influence spread computation. Aiming at higher efficiency, Borgs et al. [5] introduced a reverse reachable sketching technique (RRS) without sacrificing the accuracy guarantee. Tang et al. [28, 29] proposed the TIM/TIM+ and IMM algorithms, and Li et al. [21] designed indexing methods, all based on the RRS technique, to further improve its efficiency. However, all these works assume that the influence cascade probability between a pair of users is fixed; and unlike ours, they do not consider the impacts of selecting the top-r tags in order to maximize the influence spread.

Topic Dependent Influence Maximization. The classic influence maximization problem has been recently considered in a topic-aware fashion [2, 4, 6, 20]. In both [2, 6], topic-aware influence maximization solves a different problem, i.e., finding a set of seed nodes that maximize the spread of information for a *given* topic set. In contrast, the top-r relevant tags are not known apriori in our setting. In [3], Barbieri et al. introduced the problem of identifying the top-k seed nodes, together with finding the optimal topic distribution. The key difference with our work is that finding the top-r relevant tags is different than finding the optimal topic distribution. As we reasoned earlier, in real-world scenarios, it is generally hard to achieve a specific (optimal) topic distribution for a campaign. Instead, we provide the top-k tags directly that a campaigner can associate to maximize the spread of her influence among the target customers. In addition, we have novel technical contributions via indexing for targeted influence maximization, selecting good initial conditions, and proving the local optimality of our iterative algorithm, aiming towards scalability and a high-quality solution. Very recently, Li et al. [20] explored a user's most influential topic set in a social network. Our work is different as we aim at finding not only the top-r relevant tags, but also the top-k seed nodes for targeted influence maximization in a social network.

Conditional Reliability. Conditional reliability has been widely studied in systems and device networks[1], and is recently introduced over uncertain graphs in [13, 14]. They define reliability (i.e., the probability that a given set of source nodes can reach a given set of target nodes) when edge-existence probabilities are conditioned on external factors (i.e., catalysts), and study the problem of finding top-k catalysts that maximize the reliability between a limited number of source-target pairs. On the other hand, our objective is different: Jointly find the top-r tags and the top-k seed nodes for targeted influence maximization in a social network. For our top-r tags selection problem, we develop Batch-paths selection algorithm, following the idea of Individual-paths selection method in [13, 14]. We, however, show that Batch-paths selection method can achieve up to 30% more influence spread compared to Individual-paths selection, while the running time remains similar.

8 CONCLUSIONS

In this paper, we introduced and investigated the novel problem of jointly finding the top-k seed nodes and the top-r relevant tags for targeted influence maximization in a social network. We deeply characterized the problem, proving that it is **NP**-hard and also hard to approximate. To solve this problem, we refined both state-of-the-art seeds finding and tags finding algorithms together with indexing and batch-path selection methods, and thus improved their efficiency by 30% and accuracy up to 30%, respectively. Finally, we developed an iterative framework, together with smart initialization techniques, that alternatively optimizes the top-k seeds and the top-r tags, and we proved that the influence spread monotonically increases until convergence. Our experimental results demonstrated that the proposed iterative algorithm achieves high influence spread compared to baselines, and converges fast in 3 iterations.

REFERENCES

[1] K. K. Aggarwal, K. B. Misra, and J. S. Gupta. Reliability Evaluation A Comparative Study of Different Techniques. *Micro. Rel.*, 14(1), 1975.
[2] C. Aslay, N. Barbieri, F. Bonchi, and R. A. Baeza-Yates. Online Topic-aware Influence Maximization Queries. In *EDBT*, 2014.
[3] N. Barbieri and F. Bonchi. Influence Maximization with Viral Product Design. In *SDM*, 2014.
[4] N. Barbieri, F. Bonchi, and G. Manco. Topic-Aware Social Influence Propagation Models. In *ICDM*, 2012.
[5] C. Borgs, M. Brautbar, J. Chayes, and B. Lucier. Maximizing Social Influence in Nearly Optimal Time. In *SODA*, 2014.
[6] S. Chen, J. Fan, G. Li, J. Feng, K. L. Tan, and J. Tang. Online Topic-aware Influence Maximization. *PVLDB*, Volume 8 Issue 6:666–677, 2015.
[7] W. Chen, L. V. S. Lakshmanan, and C. Castillo. *Information and Influence Propagation in Social Networks.* Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2013.
[8] W. Chen, C. Wang, and Y. Wang. Scalable Influence Maximization for Prevalent Viral Marketing in Large-Scale Social Networks. In *KDD*, 2010.
[9] D. Eppstein. Finding the k Shortest Paths. *SIAM J. Comput*, 28(2):652–673, 1998.
[10] A. Goyal, F. Bonchi, and L. V. S. Lakshmanan. A Data-Based Approach to Social Influence Maximization. *PVLDB*, Volume 5 Issue 1:73–84, 2011.
[11] A. Goyal, W. Lu, and L. V. S. Lakshmanan. CELF++: Optimizing the Greedy Algorithm for Influence Maximization in Social Networks. In *WWW*, 2011.
[12] D. Kempe, J. M. Kleinberg, and E. Tardos. Maximizing the Spread of Influence through a Social Network. In *KDD*, 2003.
[13] A. Khan, F. Bonchi, F. Gullo, and A. Nufer. Conditional Reliability in Uncertain Graphs. https://arxiv.org/abs/1608.04474, 2017.
[14] A. Khan, F. Gullo, T. Wohler, and F. Bonchi. Top-k Reliable Edge Colors in Uncertain Graphs. In *CIKM*, 2015.
[15] A. Khan, B. Zehnder, and D. Kossmann. Revenue Maximization by Viral Marketing: A Social Network Host's Perspective. In *ICDE*, 2016.
[16] M. Kimura and K. Saito. Tractable Models for Information Diffusion in Social Networks. In *PKDD*, 2006.
[17] T. Lappas, E. Terzi, D. Gunopulos, and H. Mannila. Finding Effectors in Social Networks. In *KDD*, 2010.
[18] J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. VanBriesen, and N. Glance. Cost-effective Outbreak Detection in Networks. In *KDD*, 2007.
[19] X. Li, L. Guo, and Y. E. Zhao. Tag-based Social Interest Discovery. In *WWW*, 2008.
[20] Y. Li, J. Fan, D. Zhang, and K.-L. Tan. Discovering Your Selling Points: Personalized Social Influential Tags Exploration. In *SIGMOD*, 2017.
[21] Y. Li, D. Zhang, and K. Tan. Real-time Targeted Influence Maximization for Online Advertisements. *PVLDB*, Volume 8 Issue 10:1070–1081, 2015.
[22] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher. An Analysis of Approximations for Maximizing Submodular Set Functions. *Mathematical Programming*, 14(1):265–294, 1978.
[23] M. Potamias, F. Bonchi, A. Gionis, and G. Kollios. k-Nearest Neighbors in Uncertain Graphs. *PVLDB*, 2010.
[24] R.Motwani and P.Raghavan. *Randomized Algorithms.* Cambridge University Press, 1995.
[25] K. Saito, R. Nakano, and M. Kimura. Prediction of Information Diffusion Probabilities for Independent Cascade Model. In *KES*, 2008.
[26] M. Sieff. Why Hillary Clinton Lost Her Blue Wall. http://www.martinsieff.com/cycles-of-change/ hillary-clinton-lost-blue-wall/, 2016.
[27] C. Song, W. Hsu, and M. L. Lee. Targeted Influence Maximization in Social Networks. In *CIKM*, 2016.
[28] Y. Tang, Y. Shi, and X. Xiao. Influence Maximization in Near-Linear Time: A Martingale Approach. In *SIGMOD*, 2015.
[29] Y. Tang, X. Xiao, and Y. Shi. Influence Maximization: Near-Optimal Time Complexity Meets Practical Efficiency. In *SIGMOD*, 2014.

9 ACKNOWLEDGEMENT

The research is supported by MOE Tier-1 RG83/16 and NTU M4081678. Any opinions, findings, and conclusions in this publication are those of the authors and do not necessarily reflect the views of the funding agencies.

A PROOF OF THEOREM 4

A problem has a *Polynomial Time Approximation Scheme* (**PTAS**) if the problem admits a polynomial-time constant-factor approximation algorithm for *every* constant $\beta \in (0, 1)$. We prove Theorem 4 by contradiction, that is, if there exists at least one value of β such that, if a β-approximation algorithm for the top-r tags selection problem exists, then we can solve the Set Cover problem in polynomial time. Since Set Cover is **NP**-hard, clearly this can happen only if **P** = **NP**. The Set Cover problem is defined by a collection of subsets $\mathcal{S} = \{S_1, S_2, \ldots, S_m\}$ of a ground set $U = \{u_1, u_2, \ldots, u_n\}$. The decision version of Set Cover asks the following question: given r, is there any solution with at most r subsets from $\mathcal{S}$ that cover all elements in U?

Starting with an instance of Set Cover, we construct in polynomial time an instance of our top-r tags selection problem in the same way as in the proof of Theorem 3. If r subsets suffice to cover all elements in U in the original instance of Set Cover, the optimal solution C_1^* to our problem would have influence spread $\sigma(S, T, C_1^*) = n$. Otherwise, if no r sets cover all elements in U, C_1^* would have influence spread at most $\sigma(S, T, C_1^*) = n-1$. Assume that a polynomial-time β-approximation algorithm for the top-r tags finding problem exists, for some $\beta \in (0, 1)$. We call it "Approx". Approx would yield a solution C_1 such that $\sigma(S, T, C_1) \geq \beta\sigma(S, T, C_1^*)$. Now, consider the inequality $n - 1 < \beta n$. If this inequality has a solution for some values of β, by running Approx on the instance and checking the influence spread of this solution, one can answer Set Cover in polynomial time: a solution to Set Cover exists iff the solution given by Approx has influence spread $\geq \beta n$. Thus, to prove the theorem, we need to show that a solution to that inequality exists. However, it is easy to see that there will always be a value of $\beta \in (\frac{n-1}{n} + \epsilon, 1)$ for which $n - 1 < \beta n$ is satisfied ($\epsilon > 0$ is a very small fraction). This is a contradiction and completes the proof.

B PROOF OF THEOREM 5

In this proof, the Chernoff bounds [24] will be used, which are:

Lemma B.1. *Let X be the sum of c i.i.d. random variables sampled from a distribution $[0, 1]$ with a mean μ. For any $\delta > 0$,*

$$\Pr[X - c\mu \geq \delta \cdot c\mu] \leq \exp\left(-\frac{\delta^2}{2+\delta} c\mu\right) \tag{20}$$

$$\Pr[X - c\mu \leq \delta \cdot c\mu] \leq \exp\left(-\frac{\delta^2}{2} c\mu\right) \tag{21}$$

[29] denotes the influence spread for a size-k node set S in all nodes as $I(S)$, and the fraction of RR-sets covered by S as $F_R(S)$. Then it proves that the expectation of $F_R(S)$ is equal to the expectation of the fraction of influenced nodes in all nodes. Intuitively, in our targeted influence maximization setting, the expectation of $F_R(S)$ is equal to the expectation of the fraction of influenced nodes in target nodes. Therefore, we have:

$$\mathbb{E}[F_R(S)] = \frac{\mathbb{E}[I(S)]}{|T|} \tag{22}$$

TRS (Targeted Reverse Sketching) utilizes $F_R(S) \cdot |T|$ to estimate the influence spread $I(S)$. We prove that it is $(1-1/e-\epsilon)$-approximate solution with at least $1 - n^{-1}\binom{n}{k}^{-1}$ probability in two steps as follows.

Step 1: We prove that Inequality 8 ensures the following.

$$|F_R(S) \cdot |T| - \mathbb{E}(I(S))| < \frac{\epsilon}{2} \cdot OPT_T \tag{23}$$

holds with at least $1 - n^{-1}\binom{n}{k}^{-1}$ probability, where OPT_T is the maximum influence spread for any size-k node set S, and $\frac{\epsilon}{2}$ is the approximation ratio.

Step 2: The greedy algorithm of maximum coverage problem produces $(1-1/e)$ approximation solution [29]. We prove that by combining the aforementioned two approximation ratios $\frac{\epsilon}{2}$ and $(1-1/e)$, the final approximation ratio is $(1-1/e-\epsilon)$.

We first prove the first part. Let ρ be the probability that S intersects with a random RR-set, and $F_R(S)\cdot\theta$ can be regarded as the sum of θ i.i.d Bernoulli variables with mean ρ. Then $\rho\theta = F_R(S)\cdot\theta$. With Equation 22, we have:

$$\rho = \mathbb{E}\left[F_R(S)\right] = \frac{\mathbb{E}\left[I(S)\right]}{|T|} \tag{24}$$

With Equation 24, we can then derive Inequality 23 as follows.

$$\begin{aligned} |F_R(S)\cdot|T| - \mathbb{E}(I(S))| &< \frac{\epsilon}{2}\cdot OPT_T \\ |F_R(S)\cdot\theta - \rho\theta| &< \frac{\epsilon\theta}{2|T|}\cdot OPT_T \\ |F_R(S)\cdot\theta - \rho\theta| &< \frac{\epsilon\cdot OPT_T}{2|T|\rho}\cdot\rho\theta \end{aligned} \tag{25}$$

Let $\delta = \epsilon\cdot OPT_T/(2|T|\rho)$. By the Chernoff bounds [24], Equation 8, Equation 24, and the fact that $\rho = \mathbb{E}\left[I(S)\right]/|T| \le OPT_T/|T|$, we have:

$$\begin{aligned} &\Pr\left[|F_R(S)\cdot\theta - \rho\theta| \ge \frac{\epsilon\cdot OPT_T}{2|T|\rho}\cdot\rho\theta\right] \\ &< 2\exp\left(\frac{-\delta^2}{2+\delta}\cdot\rho\theta\right) \\ &= 2\exp\left(\frac{-\epsilon^2\cdot OPT_T^2}{8|T|^2\rho + 2\epsilon|T|\cdot OPT_T}\cdot\theta\right) \\ &\le 2\exp\left(\frac{-\epsilon^2\cdot OPT_T^2}{8|T|\cdot OPT_T + 2\epsilon|T|\cdot OPT_T}\cdot\theta\right) \\ &= 2\exp\left(\frac{-\epsilon^2\cdot OPT_T}{(8+2\epsilon)\cdot|T|}\cdot\theta\right) \le n^{-1}\cdot\binom{n}{k}^{-1} \end{aligned} \tag{26}$$

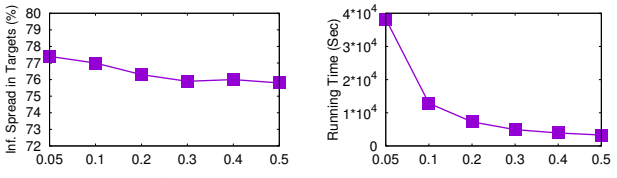
This means that Inequality 23 holds with at least $1-n^{-1}\binom{n}{k}^{-1}$ probability simultaneously for all size-k S by union bound property.

Next, we proceed to prove the second part that it is $(1-1/e-\epsilon)$-approximate solution.

Let S_k be the vertex set returned by TRS, and S_k^+ be the vertex set that maximizes $F_R(S_k^+)$ (i.e., S_k^+ intersects with the largest number of RR-sets generated by TRS). Since S_k is obtained by using $(1-1/e)$-approximate algorithm for the maximum coverage problem [29] on the RR-sets generated by TRS, we have $F_R(S_k) \ge (1-1/e)\cdot F_\theta(S_k^+)$. Let S_k^o be the actual optimal solution, i.e., $\mathbb{E}[I(S_k^o)] = OPT_T$. We have $F_R(S_k^+) \ge F_R(S_k^o)$, which leads to $F_R(S_k) \ge (1-1/e)\cdot F_R(S_k^o)$. Finally, we have:

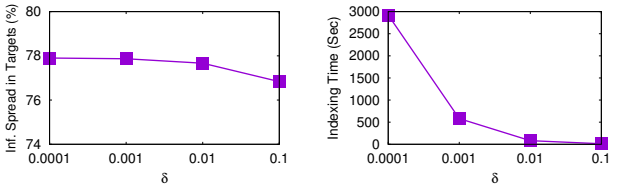
$$\begin{aligned} \mathbb{E}[I(S_k)] &> |T|\cdot F_R(S_k) - \epsilon/2\cdot OPT_T \\ &\ge (1-1/e)\cdot|T|\cdot F_R(S_k^+) - \epsilon/2\cdot OPT_T \\ &\ge (1-1/e)\cdot|T|\cdot F_R(S_k^o) - \epsilon/2\cdot OPT_T \\ &\ge (1-1/e)\cdot(1-\epsilon/2)\cdot OPT_T - \epsilon/2\cdot OPT_T \\ &> (1-1/e-\epsilon)\cdot OPT_T \end{aligned} \tag{27}$$

Thus, Theorem 5 is proved.



(a) Inf. Spread, *Twitter* (b) Running Time, *Twitter*

Figure 16: Sensitivity analysis for ϵ. $k = r = 10$, $|T| = 3K$.



(a) Inf. Spread, *Twitter* (b) Indexing Time, *Twitter*

Figure 17: Sensitivity analysis for δ. $k = r = 10$, $|T| = 3K$, $\alpha = 1$.

We estimate OPT_T by following [29], and omit the details here. The high level idea is to first generate a relatively small number of RR-sets and use them to derive an estimation of OPT_T with a bounded absolute error.

C PROOF OF LEMMA 3

The expected spread OPT_T at the current iteration is no smaller than that of the previous iteration (we proved this formally in Theorem 7). Observing Inequality 8, OPT_T is located in the denominator, which means that θ' of the current round shall be smaller than the θ of the last iteration. Next, let us consider Inequality 9, we have:

$$\begin{aligned} \theta_c - \theta_c' &= \frac{r\theta}{\alpha\delta(\theta-1)+r} - \frac{r\theta'}{\alpha\delta(\theta'-1)+r} \\ &= \frac{r\theta(\alpha\delta(\theta'-1)+r) - r\theta'(\alpha\delta(\theta-1)+r)}{(\alpha\delta(\theta-1)+r)(\alpha\delta(\theta'-1)+r)} \\ &= \frac{r^2(\theta-\theta') + r\alpha\delta(\theta'-\theta)}{(\alpha\delta(\theta-1)+r)(\alpha\delta(\theta'-1)+r)} \\ &= \frac{r(r-\alpha\delta)(\theta-\theta')}{(\alpha\delta(\theta-1)+r)(\alpha\delta(\theta'-1)+r)} \end{aligned} \tag{28}$$

Since α and δ are input parameters and are usually small (e.g., $\alpha \le 1$ and $\delta \le 0.01$), we have $r \ge \alpha\delta$ and Equation 28 will be positive. This implies that the number of indexes required in the current round will be smaller than that in previous rounds and completes the proof.

D ADDITIONAL EXPERIMENTAL RESULTS

D.1 Sensitivity Analysis

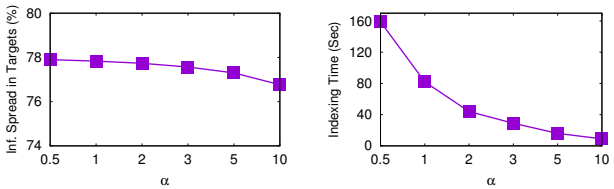
Varying ϵ. Figure 16 presents the sensitivity analysis for parameter ϵ. The running time is quite sensitive to higher ϵ, that is, every 0.1 growth in ϵ reduces about half of running time. The side-effect of higher ϵ is possible lower and unstable influence spread estimation. Therefore, we set $\epsilon = 0.1$, which is also adopted in [28, 29]

Varying δ. Figure 17 shows that with smaller δ, the algorithm returns higher accuracy, while the indexing time increases linearly. Note that when δ is small enough, e.g., 0.001 and 0.01, the accuracy remains nearly the same. Therefore, we set $\delta = 0.01$ as default.

Varying α. Figure 18 shows that the sensitivity performance for α is similar to that of δ. Referring to Equation 15, δ and α jointly decide the expected average pairwise index number, i.e., $\mathbb{E}\left[C(G)\right] = \alpha\delta$. When δ is fixed as 0.01, the accuracy remains nearly the same for

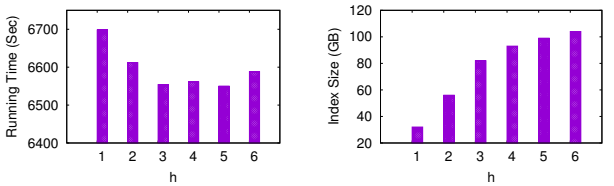
Prob. Mean	Index Size (GB), #Tags(r) = 10						Index Size (GB), #Seeds(k) = 100				Querying Time (Sec), #Tags(r) = 10											
	k=5	k=10	k=20	k=50	k=100	k=500	r=5	r=10	r=20	r=30	k=5		k=10		k=20		k=50		k=100		k=500	
											TRS	LL-TRS	TRS	LL-TRS	TRS	LL-TRS	TRS	LL-TRS	TRS	LL-TRS	TRS	LL-TRS
0.27	457	448	445	455	449	460	295	448	589	712	5697	**4752**	7504	**6497**	10946	**8687**	12363	**10299**	17017	**14694**	31004	**26484**
0.16	397	397	390	404	408	399	261	397	510	597	4416	**3312**	5615	**4660**	7918	**6809**	9221	**8022**	13060	**10366**	18699	**15053**
0.09	368	359	356	363	360	352	233	359	453	528	3547	**2873**	4240	**3095**	5783	**4800**	7428	**6091**	9946	**7759**	13041	**10824**
0.05	318	323	323	327	336	332	192	323	391	449	2758	**2289**	3229	**2389**	4095	**3276**	5450	**4633**	7812	**4306**	9301	**7992**

Table 7: Additional scalability test for LL-TRS index size and querying time (including index loading time) with varying edge probability means, seed set size k, and tag set size r, *Twitter*. $|T| = 3K$.



(a) Inf. Spread, *Twitter* (b) Indexing Time, *Twitter*

Figure 18: Sensitivity analysis for α. $k = r = 10$, $|T| = 3K$, $\delta = 0.01$.



(a) Running Time, *Twitter* (b) Index Size, *Twitter*

Figure 19: Sensitivity analysis for h. $k = r = 10$, $|T| = 3K$.

small α, e.g., 0.5, 1, and 2. We set $\alpha = 1$ as default, since (1) the indexing time decrease for $\alpha = 0.5$ to $\alpha = 1$ is more significant (Figure 18(b)); (2) we aim at avoiding the accuracy loss when $\alpha \geq 2$ as much as possible (Figure 18(a)).

Varying h. Figure 19 demonstrates that $h = 3$, 4, and 5 result in similar querying time. On the other hand, the accuracy will not be affected by h (Section 3.3). However, with higher h, clearly higher space for indexes is required (Figure 19(b)). Therefore, $h = 3$ is our default setting.

Varying Edge Probabilities. Recall that we decide edge probabilities as follows. For every edge (u, v) and a tag c, we compute the frequency (i.e., number of occurrences) of c in (u, v). Let us denote this frequency by t, then we assign probability $p((u, v)|c) = 1 - e^{-t/a}$. Now, we analyse the sensitivity for our proposed iterative algorithm towards convergence with various edge probabilities over the same graph. In particular, we obtain graphs with mean edge probability of 0.06, 0.12, 0.21, 0.33, and 0.51 by assigning a as 80, 40, 20, 10, and 5, respectively. As shown in Figure 20, the influence growing tendency and the convergence rate remain similar for a good initialization method, e.g., RS+FT. However as expected, the maximum influence spread increases for graphs with higher edge probabilities.

D.2 Additional Scalability Tests

Table 7 presents additional experimental results with respect to different edge probability means, seed set sizes (k), and tag set sizes (r). As expected, with lower edge probabilities, less edges exist in the indexes, which results in smaller index size and querying time.

We observe that larger k, in general, does not lead to higher index sizes. This is because larger k causes larger θ (Theorem 5). However, in practice, θ is always much larger than all other terms in Inequality 9. Therefore, $\alpha\delta(\theta - 1) >> r$, and $\theta_c \approx \frac{r}{\alpha\delta}$. This implies

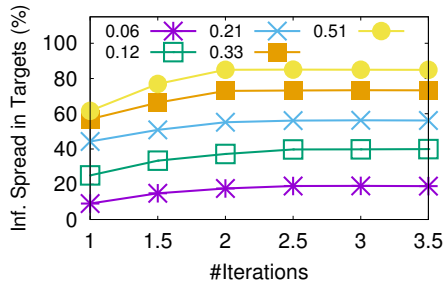


Figure 20: Influence spread after various iterations for *Yelp* graph with different edge probability means. *RS+FT* initialization method, $k = r = 20$, $|T| = 3K$.

that θ_c keeps nearly the same for different k. The slight difference of index sizes when varying k is caused by different tag sets found.

For the same reason as discussed above, θ_c tends to increase linearly with increase in r. This is also shown in Table 7 that our index sizes grow with higher values of r at a modest rate.

Clearly, our developed TRS method (without indexing) and our indexing scheme LL-TRS have trade-offs between storage and online querying time. LL-TRS improves online querying time up to 30% compared to TRS (Figure 5 and Table 7, *Twitter*), while LL-TRS has an overhead of storing indexes. We notice that our index size is modest even for the larger *Twitter* dataset. Furthermore, the index size reported in Figure 15 and Table 7 is the total size of indexes generated during the entire iterative procedure. However in a specific round, only those indexes for the chosen tags (about 30-40% of the total index size, e.g., 140 GB when $k = 100$, $r = 10$, prob. mean=0.27, *Twitter*) will be loaded into memory. This allows us to store all the indexes on disk for larger datasets (e.g., *Twitter*), while loading only the necessary indexes (i.e., indexes for the selected tags at every round) in the memory. The querying time presented in Table 7 includes the time cost of loading indexes from disk.