Reconnecting the Estranged Relationships: Optimizing the Influence Propagation in Evolving Networks

Taotao Cai
Macquarie University
Sydney, Australia
taotao.cai@mq.edu.au

Qi Lei
Chang'an University
Xi'an, China
2020024009@chd.edu.cn

Quan Z. Sheng
Macquarie University
Sydney, Australia
michael.sheng@mq.edu.au

Shuiqiao Yang
University of New South Wales
Sydney, Australia
shuiqiao.yang@unsw.edu.au

Jian Yang
Macquarie University
Sydney, Australia
jian.yang@mq.edu.au

Wei Emma Zhang
The University of Adelaide
Adelaide, Australia
wei.e.zhang@adelaide.edu.au

ABSTRACT

Influence Maximization (IM), which aims to select a set of users from a social network to maximize the expected number of influenced users, has recently received significant attention for mass communication and commercial marketing. Existing research efforts dedicated to the IM problem depend on a strong assumption: the selected seed users are willing to spread the information after receiving benefits from a company or organization. In reality, however, some seed users may be reluctant to spread the information, or need to be paid higher to be motivated. Furthermore, the existing IM works pay little attention to capture user's influence propagation in the future period. In this paper, we target a new research problem, named *Reconnecting Top-l Relationships* (RTlR) query, which aims to find l number of previous existing relationships but being estranged later, such that reconnecting these relationships will maximize the expected number of influenced users by the given group in a future period. We prove that the RTlR problem is NP-hard. An efficient greedy algorithm is proposed to answer the RTlR queries with the influence estimation technique and the well-chosen link prediction method to predict the near future network structure. We also design a pruning method to reduce unnecessary probing from candidate edges. Further, a carefully designed order-based algorithm is proposed to accelerate the RTlR queries. Finally, we conduct extensive experiments on real-world datasets to demonstrate the effectiveness and efficiency of our proposed methods.

1 INTRODUCTION

Over the past few decades, the rise of online social networks has brought a transformative effect on the communication and information spread among human beings. Through social media platforms (*e.g., Twitter*), business companies can spread their products information and brand stories to their customers, politicians can deliver their administrative ideas and policies to the public, and researchers can post their upcoming academic seminars information to attract their peers around the world to attend. Motivated by real substantial applications of online social networks, researchers start to keep a watchful eye on *information diffusion* [4, 23], as the information could quickly become pervasive through the "*word-of-mouth*" propagation among friends in social networks.

Influence Maximization (IM) is the key algorithmic problem in information diffusion research, which has been extensively studied in recent years. IM aims to find a small set of highly influential users such that they will cause the maximum influence spread in a social network [3, 23, 37, 40]. To fit with different real application scenarios, many variants of the IM problem have been investigated recently, such as *Topic-aware* IM [5, 17, 29, 30], *Time-aware* IM [14, 20, 39, 47], *Community-aware* IM [28, 42, 45, 48], *Competitive* IM [2, 32, 36, 43], *Multi-strategies* IM [7, 24], and *Out-of-Home IM* [51, 53]. However, some critical characteristics of the IM study fail to be fully discussed in existing IM works. We explain these characteristics using the two observations below.

Observation 1. Some business companies wish their product information would be spread to most of their customers in the period after they spent their budgets on their selected seed users (*e.g., Apple releases its new iPhone every September. They want to find optimal influencers in social networks to appeal to as many users as possible to purchase the new iPhone in the year ahead*). However, most of the existing IM works modelled the social networks as static graphs, while the topology of social networks often evolves over time in the real world [8, 26]. Therefore, the seed users selected currently may not give good performance for influence spread in the following time period due to the evolution of the network. To satisfy Apple's requirement, we would better predict the topology evolution of social networks in the following period and select seed users from the predicted network.

Observation 2. Existing IM studies dedicated to the influence maximization problem depend on a strong assumption – the selected seed users will spread the information. However, some of the chosen individual seed users may be unwilling to promote the product information for various reasons. Moreover, most startups and academic groups may not have the budget to motivate the seed users to spread their product or academic activities information.

Our Problem. The aforementioned observations motivate us to propose and study a novel research problem, namely Reconnecting Top-l Relationships (RTlR). Given a directed evolving graph $\mathcal{G} = \{G_i\}_0^{t-1}$, a parameter l, and an institute $\mathcal{U}$ contains a group of users, RTlR asks for reconnecting a set of l estranged relationships (*e.g., edges that have ever existed in $\mathcal{G}$ while disappearing in the near future snapshot graph G_t*). Reconnecting the selected edges in RTlR query to G_t will maximize the number of influenced users in G_t that are influenced by the members of $\mathcal{U}$.

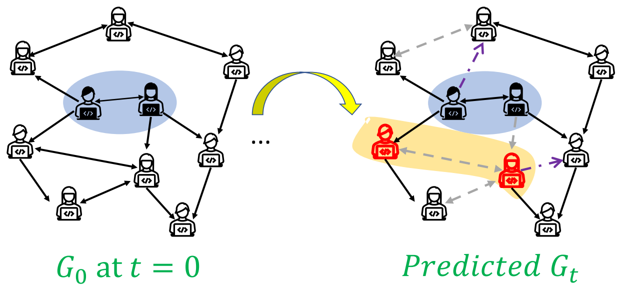


Figure 1: An example of RTlR query.

Note: the given users' group $\mathcal{U}$ are marked as black icons and covered by blue color, G_0 is the snapshot of the directed evolving graph $\mathcal{G} = \{G\}_0^{t-1}$ at time 0, and G_t is the predicted graph snapshot of $\mathcal{G}$ at time t; the greyish dotted edges in G_t represent the relationship between users exists in $\mathcal{G}$ while disappearing in G_t; the purple dotted lines represent the new adding edges in G_t; the edge of two red icons which covered by yellow color is the query result of RTlR problem.

Example 1.1 (Motivation). *LinkedIn*[1] *is a business and employment oriented online social network. It provides a social network platform to allow members to create their profiles and "connect" to each other, representing real-world professional relationships. Members can also post their activity information (e.g., employment Ads) on LinkedIn. The study of RTlR can significantly enhance the stickiness of members in LinkedIn without any budgets paid by members or LinkedIn itself.*

Figure 1 presents an evolving social network with ten members and their relationships. Suppose a research group (e.g., black icons) will host an online virtual academic seminar next month. They post the seminar information on LinkedIn because they wish to attract as many researchers as possible to join their seminar in the month ahead (e.g., G_t*). By answering the RTlR query, LinkedIn can find out the optimal estranged relationships (e.g., among the greyish dotted edges), in which reconnecting them (e.g., red icons) will maximize the spread of seminar information in the coming month. To reconnect the estranged relationships, a possible way is to send an email to the related users' platform Inbox and notify them of the recent news of their old friends. Therefore, the study of RTlR query will benefit both users and the social media platform. The members will be more willing to keep active in the network platforms, which provide them a free and efficient information post service.*

To the best of our knowledge, this is the first IM study that draws the inspiration from the intersection of (1) topology evolving prediction of social networks, and (2) no additional cost. As a result, the following challenges are important to be addressed.

Challenges. The first challenge is how to predict the topology of social networks in a specified future period. To deal with this challenge, we adopt the link prediction method [50] to predict the network structure evolution in evolving networks. The other challenge is the complexity of RTlR query problem. Unlike traditional IM studies that aim to find Top-k influential users, our RTlR focuses on the edges discovery. The existing IM algorithms are not applicable to address the RTlR query, and a more detailed analysis is presented in Section 4.1. Thirdly, our RTlR query may return different results for different given user groups, while the IM problem only needs to be queried one time to get the most influential users.

To address these algorithmic challenges, we first propose a sketched-based greedy (SBG) algorithm to answer the RTlR query of a given group. Besides, a candidate edges reducing method has been proposed to boost the SBG algorithm's efficiency. Furthermore, we carefully designed a novel order-based SBG algorithm to accelerate the RTlR query.

Contributions. We state our major contributions as follows:

- We introduce and formally define the problem of *Reconnecting Top-l Relationships* (RTlR) for the first time, and explain the motivation of solving the problem with real applications. We also prove that the RTlR query problem is NP-hard.
- We propose a sketch-based greedy (SBG) approach to answer the RTlR queries. Besides, we present the pruning method to boost the efficiency of the SBG algorithm by reducing the number of candidate edges' probing.
- To further accelerate the RTlR query, we elaborately design a novel order-based algorithm to answer the RTlR query more efficiently.
- We conduct extensive experiments to demonstrate the efficiency and effectiveness of our proposed algorithms using real-world datasets.

Organization. The remainder of this paper is organized as follows. First, we present the preliminaries in Section 2 and formally define the RTlR problem in Section 3. Then, we propose the sketch-based greedy approach and the accelerate method in Section 4. We further present a new order-based algorithm to efficiently answer the RTlR query in Section 5. After that, the experimental evaluation and results are reported in Section 6. Finally, we review the related works in Section 7 and conclude this work in Section 8.

Table 1: Frequently used notations

Notation	Definition and Description
$\mathcal{G} = \{G\}_0^{t-1}$	a directed evolving graph
G_i	the snapshot graph of $\mathcal{G}$ at time point i
V; E_i	the vertex set and edge set of G_i
G_t	the predict snapshot graph of $\mathcal{G}$ at time point t
$\mathcal{U}$	the given users group
$I(\mathcal{U}, G)$	the number of activated users in graph G by users in $\mathcal{U}$
$E(I(\mathcal{U}, G))$	the expected number of users in graph G that influenced by users set $\mathcal{U}$
θ_1	the number of generated RR sets
S (S_e)	Candidate seed users (edges) set of IM (RTlR) query problem
$G_t \oplus S_e$	Reconnecting the edges in S_e of graph G_t
OPT (OPT^*)	the maximum expected spread of any size-k seed users (edges) set of IM (RTlR) query problem
θ_2	the number of generated sketch subgraphs
$G_{sg} = \{G_{sg}^j\}_1^{\theta_2}$	the sketch subgraph set
θ_3	the number of generated sketch subgraphs in the SBG method

[1]https://www.linkedin.com/

2 PRELIMINARY

We define a directed evolving network as a sequence of graph snapshots $\mathcal{G} = \{G_i\}_0^{t-1}$, and $\{0,,1,..,t-1\}$ is a set of time points. We assume that the network snapshots in $\mathcal{G}$ share the same vertex set. Let G_i represent the network snapshot at timestamp $i \in [0, t-1]$, where each vertex u in V is a social user in G_i, each edge $e = (u, v)$ in E_i represents a cyber link or a social relationship between users u and v in G_i. Similar to [11, 21], we can create "dummy" vertices at each time step i to represent the case of vertices joining or leaving the network at time i (*e.g.,* $V = \cup_{i=1}^{t-1} V^i$ *where* V^i *is the set of vertices truly exist at* i). Besides, each edge $(u, v) \in E$ in G is associated with a *propagation probability* $p(u, v) \in [0, 1]$. Table 1 summarizes the mathematical notations frequently used throughout this paper.

2.1 Link Prediction

Link prediction is an important network-related problem firstly proposed by Liben-Nowell et al. [31], which aims to infer the existence of new links or still unknown interactions between pairs of nodes based on their properties and the currently observed links.

Given a directed evolving graph $\mathcal{G} = {G_i}_0^{t-1}$ with the time points set $\{0, 1, .., t-1\}$, in this paper, we use the recent link prediction method [49, 50], named learning from **S**ubgraphs, **E**mbeddings, and **A**ttributes for **L**ink prediction (SEAL) method, to predict the graph structure of snapshot graph G_t of $\mathcal{G}$ at the future time point t. Specifically, SEAL is a graph neural network (GNN) based link prediction method that transforms the traditional link prediction problem into the subgraph classification problem. It first extracts the h-hop enclosing subgraph for each target link, and then applies a labeling trick, called Double Radius Node Labeling (DRNL), to add an integer label for each node relevant to the target link as its additional feature. Next, the above-labeled enclosing subgraphs are fed to GNN to classify the existence of links. Finally, it returns the predicted graph G_t of evolving graph $\mathcal{G}$ at time point t.

2.2 Influence Maximization (IM) Problem

To better understand the IM problem, we first introduce the influence diffusion evaluation of given users.

The independent cascade (*IC*) model [23] is the widely adopted stochastic model which is used for modeling the influence propagation in social networks. In the IC model, for each graph snapshot G_i, the *propagation probability* $p(u, v)$ of an edge (u, v) is used to measure the social impact from user u to v. This probability is generally set as $p(u, v) = \frac{1}{d(v)}$, where $d(v)$ is the degree of v. Every user is either in an *activated* state or *inactive* state. S_0 be a set of initial *activated* users, and generates the active set S_t for all time step $t \geq 1$ according to the following randomized rule. At every time step $t \geq 1$, we first set S_t to be S_{t-1}; Each user u activated in time step t has one chance to activate his or her neighbours v with success probability $p(u, v)$. If successful, we then add v into S_t and change the status of v to *activated.* This process continues until no more possible user activation. Finally, S_t is returned as the *activated* user set of S_0.

Let $I(S, G_i)$ be the number of vertices that are activated by S in graph snapshot G_i on the above influence propagation process under the IC model. The IM problem aims to find a size-k seed set S with the maximum expected spread $E(I(S, G_i))$. We define the IM problem as follows:

Definition 2.1 (IM problem [23]). *Given a directed graph snapshot* $G_i = (V, E_i)$*, an integer* k*, the IM problem aims to find an optimal seed set* S^* *satisfying,*

$$S^* = \underset{S \subseteq V, |S|=k}{\arg\max}\ E(I(S, G_i)) \tag{1}$$

Let OPT be the maximum expected spread of any size-k seed set, then we have $OPT = E(I(S^*, G_i))$.

2.3 Reverse Reachable Sketch

The *Reverse Influence Set* (RIS) [3] sampling technique is a *Reverse Reachable Sketch-based* method to solve the *IM* problem. By reversing the influence diffusion direction and conducting reverse *Monte Carlo* sampling [25], *RIS* can significantly improve the theoretical run time bound.

Definition 2.2 (Reverse Reachable Set [3]). *Suppose a user* v *is randomly selected from* V*. The reverse reachable (RR) set of* v *is generated by first sampling a graph* g *from* G_i*, and then taking the set of users that can reach to* v *in* g*.*

By generating θ_1 RR sets on random users, we can transform the IM problem to find the optimal seed set S, while S can cover most RR sets. This is because if a user has a significant influence on other users, this user will have a higher probability of appearing in the RR sets. Besides, Tang et al. [41] proved that when θ_1 is sufficiently large, RIS returns near-optimal results with at least $1-|V|^{-1}$ probability. Therefore, the process of using the RIS method to solve the IM query contains the following steps:

1 Generate θ_1 random RR sets from G_i.
2 Find the optimal user set S which can cover the maximum number of above generated RR sets.
3 Return the user set S as the query result of IM query problem.

Theorem 2.1 (Complexity of RIS [40]). *If* $\theta_1 \geq (8 + 2\varepsilon) \cdot |V| \cdot \frac{ln|V| + ln\binom{|V|}{k} + ln2}{OPT \cdot e^2}$*, RIS returns an* $(1 - \frac{1}{e} - \varepsilon)$ *approximate solution to the IM problem with at least* $1 - |V|^{-1}$ *probability.*

2.4 Forward Influence Sketch

The Forward Influence Sketch (FI-SKETCH) method [9, 10, 35] constructs a sketch by extracting the subgraph induced by an instance of the influence process (*e.g., the IC model*). Then, it can estimate the influence spread of a seed set S using these subgraphs accurately with theoretical guarantee. The process of using the FI-SKETCH method to solve the IM query contains the following steps:

1 Generate θ_2 sketch subgraph G_{sg}^j by removing each edge $e = (u, v)$ from G_i with probability $1 - P_{u,v}$.
2 Find the optimal user set S, while the average number of users reached by S within θ_2 constructed sketches graphs is maximum.
3 Return the user set S as the query result of IM query problem.

Theorem 2.2 (Complexity of FI-SKETCH [9]). *If* $\theta_2 \geq (8 + 2\varepsilon) \cdot |V| \cdot \frac{ln|V| + ln\binom{|V|}{k} + ln2}{\varepsilon^2}$*, FI-SKETCH returns an* $(1 - \frac{1}{e} -$

ε) *approximate solution to the IM problem with at least* $1 - |V|^{-1}$ *probability.*

3 PROBLEM DEFINITION

In this section, we formulate the *Reconnecting Top-l Relationships* (RTlR) query problem and analyze its complexity.

Definition 3.1 (RTlR Problem). *Given a directed evolving graph* $\mathcal{G} = \{G_i\}_0^{t-1}$*, the parameter* l*, and a group of users* $\mathcal{U}$*, the problem of Reconnecting Top-l Relationships (RTlR) asks for finding an optimal edge set* S *with size* l *in predicted graph snapshot* G_t *of* $\mathcal{G}$ *at time* t*, where the expected spread of* $\mathcal{U}$ *will be maximized while reconnecting edges of* S_e *in* G_t *(e.g.,* $\widehat{G_t} = G_t \oplus S_e$*). Formally,*

$$\widehat{S_e} = \underset{S_e \subseteq \mathcal{G} \backslash G_t}{\arg\max}\, E(I(\mathcal{U}, \widehat{G_t}) \tag{2}$$

In the following, we conduct a theoretical analysis on the hardness of the RTlR problem.

Theorem 3.1 (Complexity). *The RTlR problem is NP-hard.*

Proof. We prove the hardness of RTlR problem by a reduction from the decision version of the maximum coverage (MC) problem [22]. Given an integer l and several sets where the sets may have some elements in common, the maximum coverage problem aims to select at most l of these sets to cover the maximum number of elements. Furthermore, we need to discuss the existence of a solution that the MC problem is reducible to the RTlL problem in polynomial time.

Given a directed evolving graph $\mathcal{G}$, a group of users $\mathcal{U}$, and the predicted snapshot graph G_t from $\mathcal{G}$, we reduce the MC problem to RTlL with the following process: (1) For a given group $\mathcal{U}$, we compute the influence users set of $\mathcal{U}$ as $I(\mathcal{U}, G_t)$; (2) $\forall e \in \mathcal{G} \backslash G_t$, we create a set S_e with the elements collected from the influenced users $I(\mathcal{U}, \widehat{G_t}) - I(\mathcal{U}, G_t)$ while $\widehat{G_t} = G_t \oplus e$; (3) We set the reconnecting edges of RTlL as l, which is the same as the input of *MC*. The above reduction can be done in polynomial time. Since the *Maximum Coverage* problem is NP-hard, so is the RTlL problem. □

Theorem 3.2 (Influence Spread). *The influence spread function* $I(.)$ *under the RTlR problem is monotone and submodular.*

Proof. Given a snapshot graph G_t, and a group $\mathcal{U} \in V(G_t)$, $I(\mathcal{U}, G_t)$ represents the influenced user set of $\mathcal{U}$. For two edge sets $S_e \subseteq T_e$, we have $I(\mathcal{U}, G_t \oplus S_e) \leq I(\mathcal{U}, G_t \oplus T_e)$. Then, we have verified that $I(.)$ is *monotone*. Besides, for a new reconnecting edge e, the marginal contribution when added to set S_e and T_e respectively satisfies $I(\mathcal{U}, G_t \oplus (S_e \cup e)) - I(\mathcal{U}, G_t \oplus S_e) \geq I(\mathcal{U}, G_t \oplus (T_e \cup e)) - I(\mathcal{U}, G_t \oplus T_e)$. Therefore, we have proved that $I(.)$ is *submodular*. Thus, we can conclude that the influence spread function $I(.)$ of RTlL problem is *monotone* and *submodular*. □

4 SKETCH BASED GREEDY ALGORITHM

To answer the RTlR query problem, we first predict the graph structure of the given evolving graph $\mathcal{G}$ at t by using the link prediction method [50]. According to Theorem 3.2, the influence spread function of RTlR is *submodularity* and *monotonicity*. Therefore, one possible solution of the RTlL problem is to use the greedy approach to iteratively find out the most influential edge e, in which reconnecting e in predicted snapshot graph G_t will maximize the influence spread of given users group $\mathcal{U}$ in $\widehat{G_t}$ (*e.g.,* $\widehat{G_t} = G_t \oplus e$). So far, the remaining challenge of RTlR query is to evaluate the effect of a reconnected edge e on the influence spread of $\mathcal{U}$ in G_t.

4.1 Existing IM Approaches Analysis

As mentioned in [23], we can estimate the influence spread of given users by using the *Monte Carlo* simulation. Specifically, given users group $\mathcal{U}$, we simulate the randomized diffusion process with $\mathcal{U}$ in G_t for $\mathcal{R}$ times. Each time we count the number of active users after the diffusion ends, and then we take the average of these counts over the $\mathcal{R}$ times as the estimated number of influenced users of $\mathcal{U}$. However, the *Monte Carlo* simulation method is much time-consuming and cannot be used in the large graph. Later on, Borgs et al. [3] proposed a *Reverse Reachable Sketch-based* method to the *IM* problem, named *Reverse Influence Set* (RIS) sampling, and the extended versions of the RIS method [33, 34, 40] were widely used to answer the IM problem as the state-of-the-art IM query methods. The *Reverse Influence Set* (RIS) sampling technique is a *Reverse Reachable Sketch-based* method to the IM problem. By reversing the influence diffusion direction and conducting reverse *Monte Carlo* sampling, *RIS* can significantly improve the theoretical run time bound of the IM problem.

Unfortunately, the RIS sampling method is not suitable for answering our RTlR query. That is because the RIS sampling is designed to find the Top-k most influential users in a graph, but our RTlR query focuses on reconnecting several optimal edges to enhance a given user group's influence spread. In particular, the RIS sampling method transforms the IM problem to find the optimal seed set S by generating θ_1 RR sets, while S can cover most RR sets. The RR sets only contain the user's information while discarding the graph sketch (*e.g., the edge's information*). Therefore, if we use the RIS sampling to answer the RTlR query, we have to recompute the RR sets for each edge insertion during the RTlR query process, which is time-consuming and unrealistic in large graphs.

4.2 FI-Sketch based Greedy Algorithm

Facing the challenges mentioned above, we propose a sketch-based greedy (SBG) method to answer the RTlR query. Precisely, we first set θ_3 as a sufficient number of generated sketch subgraphs in our SBG method to theoretically ensure the quality of the returned results for the RTlR query (*i.e., the details of how* θ_3 *should be set will further discuss in Section 4.3*). Then, we use the FI-SKETCH to evaluate the effect of a new adding edge e on the influence spread of a given users group $\mathcal{U}$ based on the θ_3 generated sketch subgraphs. Compared with the RIS approach, the graph structure information was contained in the generated θ_3 sketch subgraphs during the process of the FI-SKETCH approach (refer to Section 2.4), so that we do not need to recompute the sketches while the edges update.

The details of the SBG method are described in Algorithm 1. In the pre-computing phase (Lines 1-3), we predict the snapshot graph G_t using the link prediction method [50], and then generate θ_3 random sketch graphs by removing each edge $e = (u, v)$ from G_t with probability $1 - P_{u,v}$. Besides, based on Definition 3.1, we initialize $CE \in \{\mathcal{G} \backslash G_t\}$ as the candidate edges set of the RTlR query.

Algorithm 1: RT*l*R: SBG

Input: $\mathcal{G} = \{G_i\}_0^{t-1}$: an evolving graph, l: the number of selected edges, and $\mathcal{U}$: a group of users

Output: $\widehat{S_e}$: the optimal reconnecting edge set

```
1  Predict the snapshot graph G_t from 𝒢 [50];
2  Generate θ_3 sketch subgraph G_sg = {G_sg^j}_1^{θ_3};
3  Initialize Ŝ_e ← ∅, Candidate edges set CE ∈ {𝒢 \ G_t};
4  for i = 1 to l do
5  |   ê ← arg max_{e∈CE} FI-SKETCH(𝒰, e);
6  |   Ŝ_e ← Ŝ_e ∪ ê;
7  return Ŝ_e
8  Function FI-SKETCH(𝒰, e):
9  |   count ← 0;
10 |   for j = 1 to θ_3 do
11 |   |   Ĝ_sg^j ← G_sg^j ⊕ {Ŝ_e ∪ e};
12 |   |   n_a ← the number of vertexes reached by 𝒰 in Ĝ_sg^j;
13 |   |   count ← count + n_a;
14 |   return count / θ_3
15 End Function
```

In the main body of SBG (Lines 4-6), we use the greedy method to iteratively find the l number of optimal reconnecting edges. Specifically, in each iterative, we call the *FI-SKETCH Function* to find out the optimal edge $\widehat{e}$ from the candidate edge set CE and add $\widehat{e}$ into set $\widehat{S_e}$, while reconnecting the selected edge can maximize the influence diffusion of given users group $\mathcal{U}$. Meanwhile, given an edge e, the *FI-SKETCH Function* returns back the influenced users evaluation results by using the *Forward Influence Sketch* method mentioned in Section 2.4 (Lines 8-14). Finally, we return edges set $\widehat{S_e}$ as the result of RT*l*R query (Line 7).

Complexity. The time complexity of calling the FI-SKETCH function for each candidate edges is $O(\theta_3 \cdot |E_t|)$, while the space complexity is $O(\theta_3 \cdot (|V| + |E_t|))$. Hence, the time complexity and space complexity of SBG algorithm are $O(l \cdot |CE| \cdot \theta_3 \cdot |E_t|)$ and $O(\theta_3 \cdot (|V| + |E_t|))$, respectively.

4.3 Theoretical Analysis of SBG

In this part, we will establish our theoretical claims for SBG. Specifically, we analyze how θ_3 should be set to ensure our SBG method returns near-optimal results to RT*l*R query with high probability. Our analysis highly relies on the *Chernoff bounds* [19].

Lemma 4.1. *Let $X_1,...,X_r$ be r number of independent random variables in $[0, 1]$ and $X =$ with a mean μ. For any $\sigma > 0$, we have*

$$\begin{aligned} Pr[X - r\mu \geq \sigma \cdot r\mu] &\leq exp(-\frac{\sigma^2}{2+\sigma} r\mu), \\ Pr[X - r\mu \leq -\sigma \cdot r\mu] &\leq exp(-\frac{\sigma^2}{2} r\mu). \end{aligned} \tag{3}$$

Let $\mathcal{U}$ be a group of users, S_e be the selected reconnecting edges, $\mathcal{R}_2$ be the number of generated sketch subgraphs in the SBG algorithm (*Algorithm 1*), and $F_R(\mathcal{U}, S_e)$ be the total number of additional reached users by $\mathcal{U}$ in each sketch subgraph after reconnecting edges in S_e. From [35], the expected value of $\frac{F_R(\mathcal{U},S_e)}{\mathcal{R}_2}$ equals the expected influence diffusion enhance by reconnecting edges of S_e in G_t. Then, we have the following lemma.

Lemma 4.2. $E[\frac{F_R(\mathcal{U},S_e)}{\mathcal{R}_2}] = E[I(\mathcal{U}, G_t \oplus S_e) - I(\mathcal{U}, G_t)]$

Proof. Each sketch subgraph in the SBG algorithm is generated by removing each edge e with $1-p(e)$ probability. From [35], we can observe that the expected value of the average number of reached users to $\mathcal{U}$ in all sketch subgraphs is equal to the expected spread of $\mathcal{U}$ in G_t. From the above relation of equality, we can easily deduce that $E[\frac{F_R(\mathcal{U},S_e)}{\mathcal{R}_2}] = E[I(\mathcal{U}, G_t \oplus S_e) - I(\mathcal{U}, G_t)]$. □

Theorem 4.1 (Approximate ratio). *By generating θ_3 sketch subgraphs with $\theta_3 \geq (8 + 2\varepsilon) \cdot |V| \cdot \frac{ln|V| + ln\binom{|V|}{l} + ln2}{\varepsilon^2}$, we have $|\frac{F(\mathcal{U},S_e)}{\theta_3} - (E[I(\mathcal{U}, G_t \oplus S_e) - I(\mathcal{U}, G_t)])| < \frac{\varepsilon}{2}$ holds with probability $1 - |V|^{-l}$ simultaneously for all selected edges set S (i.e., $|S| = l$).*

Proof. We can prove Theorem 4.1 by tweaking the proof in Theorem 2.1 of [40]. Let ρ be the probability of $\mathcal{U}$ can activate a fixed user v after reconnecting edges in S_e in G_t. Based on Lemma 4.2,

$$\rho = E[\frac{F_R(\mathcal{U}, S_e)}{\mathcal{R}_2}]/|V| = (E[I(\mathcal{U}, G_t \oplus S_e) - I(\mathcal{U}, G_t)])/|V| \tag{4}$$

Then, we have

$$\begin{aligned} &Pr[|\frac{F_R(\mathcal{U}, S_e)}{\theta_3} - (E[I(\mathcal{U}, G_t \oplus S_e) - I(\mathcal{U}, G_t)])|] \geq \frac{\varepsilon}{2} \\ &= Pr[|\frac{F_R(\mathcal{U}, S_e)}{|V|} - \rho\theta_3|] \geq \frac{\varepsilon\theta_3}{2|V|} \end{aligned} \tag{5}$$

Let $\sigma = \frac{\varepsilon}{2|V|\rho}$. Based on Lemma 4.1, we have

$$\begin{aligned} Equation\ (5) &< 2 \cdot exp(-\frac{\sigma^2}{2+\sigma} \cdot \rho \cdot \theta_3) \\ &= 2 \cdot exp(\frac{\varepsilon^2}{8|V|^2\rho + 2|V|\varepsilon} \cdot \theta_3) \\ &\leq 2 \cdot exp(-\frac{\varepsilon^2}{8|V| + 2\varepsilon|V|} \cdot \theta_3) \\ &\leq \frac{1}{|V|^l}. \end{aligned} \tag{6}$$

Thus, Theorem 4.1 is proved. □

Theorem 4.2 (Complexity of SBG). *With a probability of $1 - |V|^{-l}$, the SBG method for solving the RTlR query problem requires $\theta_3 \geq (8 + 2\varepsilon) \cdot |V| \cdot \frac{ln|V| + ln\binom{|V|}{k} + ln2}{\varepsilon^2}$ number of sampling sketch subgraphs so that an $(1 - \frac{1}{e} - \varepsilon)$ approximation ration is achieved.*

Proof. The proof of Theorem 4.2 is summarized as following three steps. Firstly, based on the property in Theorem 4.1, if the number of generated sampling sketch subgraphs $\theta_3 \geq (8+2\varepsilon) \cdot |V| \cdot \frac{ln|V| + ln\binom{|V|}{k} + ln2}{\varepsilon^2}$, then we have $|\frac{F(\mathcal{U},S_e)}{\theta_3} - (E[I(\mathcal{U}, G_t \oplus S_e) - I(\mathcal{U}, G_t)])| < \frac{\varepsilon}{2}$ holds with probability $1 - |V|^{-l}$. Secondly, the SBG method we proposed in this paper to solve the RT*l*R problem by utilizing the greedy algorithm of *maximum coverage* problem [22], which produces a $(1-\frac{1}{e})$ approximation solution (*mentioned in Theorem 3.1*). Finally, by combining the above two approximation ration $\frac{\varepsilon}{2}$ and $(1-\frac{1}{e})$, we can conclude the final approximation ration of our SBG method for solving RT*l*R query problem is $(1-\frac{1}{e}-\varepsilon)$ with at least $1-|V|^{-l}$ probability. □

4.4 Reducing # Candidate Edges

Since the SBG algorithm's time complexity is cost-prohibitive, which would hardly be used for dealing with the sizeable evolving graph. In this subsection, we present our optimization method by pruning the unnecessary potential edges in candidate edge set CE. The core idea behind this optimization strategy is to eliminate the edges in CE which will not have any benefit to expend the influence spread of given users group $\mathcal{U}$ while reconnecting it.

We use the symbol $u \leftsquigarrow \mathcal{U}$ to denote that u can be reached by $\mathcal{U}$. In order to reduce the size of CE, we present the below theorem to identify the quality reconnecting edge candidates (denote as $\widehat{CE}$) from CE.

Theorem 4.3 (Reachability). *Given a directed snapshot graph G_t and a users group $\mathcal{U}$, if an edge $e=(u,v)$ is selected to reconnect, one of its related users (i.e., u or v) requires to be reached by $\mathcal{U}$ in G_t; that is $e \in \widehat{CE}$ implies $u \leftsquigarrow \mathcal{U}$ or $v \leftsquigarrow \mathcal{U}$ in G_t.*

Proof. We prove the correctness of this theorem by contradiction. The intuition is that at least one pathway exists from a user to all of its influenced users in social networks. For the selected edge $e=(u,v)$, if both the user u and v are not reached by the users group $\mathcal{U}$, then the pathway between $\mathcal{U}$ and e does not exist. Therefore, reconnecting the edge e does not bring any benefits to the expansion of influence spread starting from $\mathcal{U}$, which contradicts with Definition 3.1. Thus, the theorem is proved. □

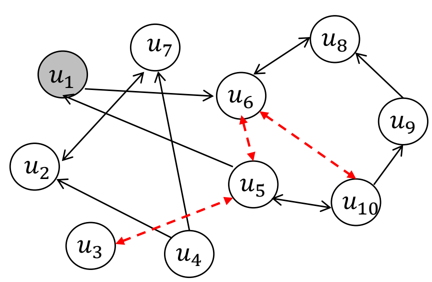


Figure 2: Running Example

Example 4.1. *Figure 2 shows a snapshot graph G_t with 10 nodes and 9 edges. The candidate edges set of RTlR is $CE=\{(u_3,u_4),(u_5,u_6),(u_6,u_{10})\}$. For a given user group $\mathcal{U}=\{u_1\}$, the pruned candidate edge set would be $\widehat{CE}=\{(u_5,u_6),(u_6,u_{10})\}$ due to $u_6 \leftsquigarrow \mathcal{U}$.*

Based on Theorem 4.3, we present a BFS-based method for pruning the candidate edge set CE in graph G_t with a given users group $\mathcal{U}$. The core idea of the BFS-based algorithm is to traverse the graph G_t starting from the nodes in $\mathcal{U}$ by performing breadth-first search (BFS). For edges in CE, if both of its related nodes are not visited in the above BFS process, then we directly prune it.

Algorithm 2: Reducing CE # BFS $(CE, \mathcal{U})$

```
1  Initialize set ĈE ← ∅, an empty Queue Q;
2  Initialize visited array A with size |V| as FALSE;
3  for each u ∈ U do
4      A[u] = TRUE;
5      Enqueue u into Q;
6  while Q is not empty do
7      Dequeue v from Q;
8      for each neighbor v′ ∈ nbr(v, G_t) in G_t do
9          if A[v′] = FALSE then
10             A[v′] = TRUE, Enqueue v′ into Q;
11         else
12             Continue;
13 for e = (u, v) ∈ CE do
14     if A[u] = TRUE or A[v] = TRUE then
15         add e into ĈE
16     else
17         continue;
18 return ĈE
```

In Algorithm 2, we outline the major steps of the BFS-based method for processing the CE pruning. Initially, each user u in graph G_t are marked a visiting status as FALSE (Line 2). Then, for the users in a given group $\mathcal{U}$, we update its visiting status as TRUE (Lines 3-5). Further, we process a BFS search starting from root user $v \in \mathcal{U}$, and update the status of each visited users as TRUE (Lines 6-12). Next, based on Theorem 4.3, we reduce all candidate edges $e=(u,v)$ from CE while both u and v have the FALSE visited status (Lines 13-17), and finally, we return the pruned candidate edges set $\widehat{CE}$ (Line 18).

Complexity. Obviously, for a given group $\mathcal{U}$, the time complexity of Algorithm 2 is $O(|V|+|E_t|+|CE|)$, and the space complexity is $O(|V|)$. Furthermore, the occupied space by Algorithm 2 will be released after the pruned candidate edges $\widehat{CE}$ is returned. For each RT*l*R query with a new given users group $\mathcal{U}$ as input, we need to recall the BFS-based pruning method to reduce the size of candidate set CE with time cost $O(|V|+|E_t|+|CE|)$, which is the main drawback of the BFS-based pruning method.

5 THE IMPROVEMENT ALGORITHM

Although the SBG algorithm and its optimization method can successfully answer the RT*l*R query problem, it is still time-consuming to handle the sizeable social networks. To address this limitation, in this section, we propose an ordered sketch-based greedy algorithm, which can significantly reduce the number of edges influence probing at each iterative of RT*l*R query process, so as to answer the RT*l*R query more efficiently.

Algorithm 3: Build UBL($\mathcal{L}$, G_{sg})

1 $(\mathcal{L}, G_{sg}) \leftarrow (\emptyset, \emptyset)$;
2 Generate θ_3 sketch subgraph $G_{sg} = \{G_{sg}^{j}\}_{1}^{\theta_3}$;
3 **for** *each edge* $e = (u, v) \in CE$ **do**
4 $\quad UB_1(e) \leftarrow$ the number of vertices that can be reached from v in G_t;
5 $\quad flag(e) \leftarrow 0$;
6 $\quad$ add $(UB_1(e), flag(e))$ into $\mathcal{L}$;
7 Store $(\mathcal{L}, G_{sg})$

5.1 Algorithm Overview

Let $\mathcal{G} = \{G_0, G_1, ..., G_{t-1}\}$ be an evolving graph. We first use the temporal link prediction method [56] to predict the future snapshot of graph G_t, and the potential reconnecting edges will be selected from candidate edges set $CE = \{\mathcal{G} \setminus G_t\}$. Before introducing the core idea of our Order-based SBG algorithm, we first briefly review using the SBG algorithm to answer the RT*l*R query and analyze the bottleneck of the SBG algorithm.

For each given users group $\mathcal{U}$, the SBG algorithm aims to find l reconnecting edges by iteratively probing each edge in CE to find out the edge $\widehat{e}$ in which reconnecting $\widehat{e}$ will bring the maximum benefits to the influence spread of $\mathcal{U}$. The time complexity of influence spread by reconnection of an edge is $O(\theta_3 \cdot |E_t|)$, which is the bottleneck of the SBG algorithm.

To deal with the above limitation of the SBG method, we propose an Order-based SBG algorithm, which focuses on reducing the number of edges probing in each iteration by using our elaboratively designed two-step bounds approach together with the order-based probing strategy. Specifically, we first generate a label index (UBL) to store the first step upper bound of influence spread expansion for each candidate edge $e \in CE$ *w.r.t* $UB_1(e)$ (in Section 5.2). Then, we generate the initial second-step upper bound (UB_2) for e (*i.e.,* $UB_2.e$) from $UB_1(e)$ of the UBL index. Next, in the influence spread expansion estimation query processing of each given users group $\mathcal{U}$ and probing edge e, we narrow the second-step upper bound of e and update the $UB_1(e)$ value of UBL index, while the narrowed second-step upper bound will be served the optimal edge finding in the following iterations (in Section 5.3). Finally, we order the candidate edges by their UB_2 values. The edge probing at the current iteration will be early terminated while the second upper bound of probing edge e is less than the present influence spread expansion estimation value (in Section 5.4).

5.2 Upper Bound Label (UBL) Construction

This section introduces how to build the label index (UBL) for each candidate edge. The UBL index contains two parts, including (1) the θ_3 sketch subgraphs G_{sg}; (2) the first-step bound $UB_1(e)$ of each candidate edge e and its updating status. The details of UBL construction procedure is shown in Algorithm 3.

From Section 2.4, we first generate θ_3 sketch subgraphs from the predicted snapshot graph G_t that will be used for the future influence spread estimation (Line 2). Then, for each candidate edge e in CE, we initialize its updating mark (*i.e.,* $flag(e)$) as 0. Meanwhile, we compute the number of vertices in G_t that can be reached from e as the first step upper bound of e, denoted as $UB_1(e)$ (Lines 3 - 6). Finally, we store the Labeling Scheme $(\mathcal{L}, G_{sg})$ for RT*l*R query processing (Line 7).

Complexity. The time complexity of sketch subgraphs generation is $O(\theta_3 \cdot |E_t|)$, and the UB_1 labeling construction of all candidate edges in CE is $O(|CE| \cdot |E_t|)$. Therefore, the time complexity of UBL construction is $O(\theta_3 \cdot |E_t| + |CE| \cdot |E_t|)$. Besides, the space complexity of UBL index construction is $O(\theta_3 \cdot (|V| + |E_t|) + |CE|)$, while storage sketch subgraphs G_{sg} has space complexity $O(\theta_3 \cdot (|V| + |E_t|))$ and generating UB_1 labeling of edges in CE has space complexity of $O(|CE|)$.

5.3 Influence Spread Expanding Estimation

Here, we present the influence spread expansion estimation of given users group $\mathcal{U}$ and edge e. Further, we also introduce the strategies of narrowing the two-step upper bounds of e (i.e., $UB_1(e)$ and $UB_2(e)$) during the above estimation process.

Algorithm 4: Sketch-Estimate Function

1 **Function** `Sketch-Estimate`($\mathcal{U}, e$):
2 $\quad count \leftarrow 0$, $count_R \leftarrow 0$, $e = (u, v)$;
3 $\quad$ **for** $k = 1$ *to* θ_3 **do**
4 $\quad\quad$ **while** $SG[k][u] == 1$ && $SG[k][v] == 0$ **do**
5 $\quad\quad\quad n_a \leftarrow |\{u' \in V | u' \rightsquigarrow e \text{ in } G_{sg}^{k} \wedge SG[k][u'] == 0\}|$;
6 $\quad\quad\quad count \leftarrow count + n_a$;
7 $\quad\quad$ **if** $\mathcal{L}.flag(e) == 0$ **then**
8 $\quad\quad\quad n_R \leftarrow |\{u' \in V | u' \rightsquigarrow e \text{ in } G_{sg}^{k}\}|$;
9 $\quad\quad\quad count_R \leftarrow count_R + n_R$;
10 $\quad\quad$ **else**
11 $\quad\quad\quad$ continue;
12 $\quad$ update $(e, UB_2.e) \leftarrow (e, count/\theta_3)$ of Q;
13 $\quad$ **if** $\mathcal{L}.flag(e) == 0$ **then**
14 $\quad\quad$ update $(UB_1(e), flag(e)) \leftarrow (count_R/R, 1)$ of $\mathcal{L}$;
15 $\quad\quad \mathcal{L}.flag(e) \leftarrow 1$;
16 $\quad$ **return** $count/\theta_3$
17 **End Function**

The details of the influence spread expansion estimation are described in Algorithm 4. For a given users group $\mathcal{U}$ and edge $e = (u, v)$, the Sketch-Estimate Function aims to compute the incremental of $\mathcal{U}$'s influence spread while reconnecting edge e in graph G_t. It takes sketch subgraphs G_{sg}, query edge e and group $\mathcal{U}$, influenced marking array SG, two-step bound UB_1 and UB_2, and returns the influence spread expansion value of e to $\mathcal{U}$. We initialize two variable $count$ and $count_R$ as 0 (Line 2). Then, an inner loop fetches the total number of the reached nodes v for e in each sketch subgraph $G_{sg}^{k} \in G_{sg}$ but not be reached by $\mathcal{U}$ (*i.e.,* $SG[k][v] == 0$), and we use $count$ to record it (Lines 3-6). Meanwhile, if the first step upper bound of e is never updated (*i.e.,* $\mathcal{L}.flag(e) == 0$), we further compute the total number of nodes reached by e in each sketch graph of G_{sg}, and store the result in $count_R$ (Lines 7 - 11). Next, we

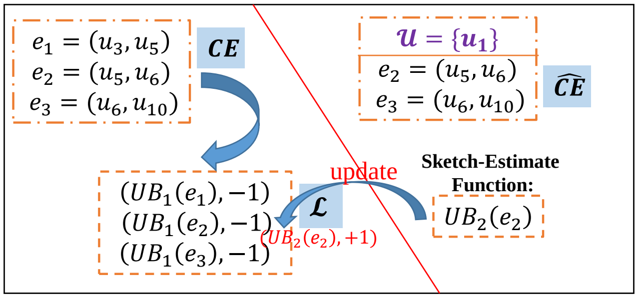


Figure 3: The Two-Step-Bounds Example

update the value of UB_2 as $count/\theta_3$, which is also the influence spread expansion value of e (Line 12); we also update $UB_1(e)$ and remark $flag(e) = 1$ of $\mathcal{L}$ when the original mark $\mathcal{L}.flag(e) = 0$ (Line 13 - 15). Finally, the influence spread expansion of e is returned (Line 16). It is remarkable that with the increasing number of RTlR queries for different given users group $\mathcal{U}$, the more edges' first-step upper bound UB_1 will be narrowed, so as to the performance of the later RTlR query with new users group will increase with no additional cost.

Complexity. It is easy for us to derive that the time complexity and space complexity of Algorithm 4 are $O(\theta_3 \cdot |E_t|)$ and $O(\theta_3 \cdot |V|)$, respectively.

Example 5.1. *Figure 3 shows a running example of our two step bounds generation. For a given graph G_t in Figure 2, we first identify the candidate edges set $CE = \{(u_3, u_5), (u_5, u_6), (u_6, u_{10})\}$. Then, we compute the first-step bound of each edge in CE (e.g., $UB_1(u_3, u_5)$) and set its initial flag as -1. During the process of each RTlL query with different given users group $\mathcal{U}$, we will prune the candidate edges set from CE to $\widehat{CE}$, and call the Sketch-Estimate Function (e.g., Algorithm 4) to estimate the influence spread expansion of each probing edge (e.g., $e_2 = (u_5, u_6)$) from $\widehat{CE}$. Meanwhile, during the above process, we get a byproduct of e_2, the second step upper bound $UB_2(e_2)$, which can be used to narrow the first upper bound of e_2 (e.g., $UB_1(e_2) \leftarrow UB_2(e_2)$). Once $UB_1(e_2)$ is updated, e_2's flag also needs to be changed to $+1$.*

5.4 Order-based SBG for RTlR Query Processing

In the previous parts of this section, we have overviewed the main idea of our order-based SBG algorithm. We also have introduced the details of two essential blocks of our Order-based SBG algorithm: (i) the UBL construction and (ii) the Sketch-Estimation Function. In the rest of this section, we will discuss the details of the Order-based SBG algorithm.

The details of the Order-based SBG algorithm are described in Algorithm 5. It takes an integer l, a users group $\mathcal{U}$, the candidate edges CE, and UBL index $(\mathcal{L}, G_{sg})$ as inputs, and returns a set $\widehat{S}$ of l optimal reconnecting edges that maximizes the influence spread of $\mathcal{U}$. We initialize a set $\widehat{S}$ as empty, an empty Priority queue Q that will be used to store the UB_2 information of candidate edges related to $\mathcal{U}$, and an array SG to mark whether a node can be reached by $\mathcal{U}$ or edges in $\widehat{S}$ at each sketch subgraphs G_{sg} (Line 1). Then, we reduce the candidate edges from CE by using Algorithm 2, and record the reduced candidate edges into set $\widehat{CE}$ (Line 2). For each edge e in $\widehat{CE}$, we get e's first-step upper bound $UB_1(e)$ from UBL index, and set $UB_1(e)$ as the initial second-step upper bound value of e (*i.e.*, $UB_2(e) = \mathcal{L}.UB_1(e)$), and then push $(e, UB_2(e))$ into priority queue Q (Lines 3 - 16). Next, we mark the nodes which are reached by $\mathcal{U}$ in each sketch subgraphs of G_{sg} (Lines 5 - 7). Further, in each iteration, we probe the candidate edges in priority queue Q in order based on their UB_2 value, and then call Sketch-Estimation Function to compute the influence spread expansion of the probing edge e, the edge probing in this iteration will be early terminated once the front edge from Q is less than the currently maximum influence spread expansion value (Line 8). After finding out the optimal edge $\widehat{e}$, we update the mark of nodes reached by $\widehat{e}$ in each sketch subgraphs (Lines 13 -17). Finally, it returns the optimal reconnecting edge set $\widehat{S}$ having maximum influence spread expansion of $\mathcal{U}$ (Line 25).

Complexity. The time complexity of Algorithm 5 is $O(|\widehat{CE}| + \theta_3 \cdot |E_t| + l \cdot |\widehat{CE}| \cdot \theta_3 \cdot |E_t|)$. Besides, the space complexity is $O(|CE| + \theta_3 \cdot |E_t|)$. Although the time complexity of Algorithm 5 is not significantly better than the SBG algorithm, it can greatly reduce the

Algorithm 5: RTlR: Order-based SBG

Input: l: the number of selected edges, $\mathcal{U}$: users group, $CE = \mathcal{G} \setminus G_t$: candidate edges, and $(\mathcal{L}, G_{sg})$: UBL

Output: $\widehat{S}$: the optimal Reconnecting edge set

1 Initialize $\widehat{S} \leftarrow \emptyset$, Priority queue Q, and Array $SG[\theta_3][|V|]$;
2 $\widehat{CE} \leftarrow$ Reducing CE # BFS (CE, $\mathcal{U}$); /*using Algorithm 2 */
3 **for** *each edge* $e = (u, v) \in \widehat{CE}$ **do**
4     $UB_2.e \leftarrow \mathcal{L}.UB_1(e)$; push $(e, UB_2.e)$ into Q;
5 **for** $i = 1$ *to* θ_3 **do**
6   **for** *each* $u \leftrightsquigarrow \mathcal{U}$ *in* G^i_{sg} **do**
7         $SG[i][u] \leftarrow 1$;
8 **for** $j = 1$ *to* l **do**
9     $(e', UB_2.e') \leftarrow Q.front$; $I_{max} \leftarrow 0$; $\widehat{e} \leftarrow e'$;
10   **while** $I_{max} < UB_2.e'$ **do**
11         $I_{max} \leftarrow$ Sketch-Estimate$(\mathcal{U}, e')$;
12         $\widehat{e} \leftarrow e'$; $(e', UB_2.e') \leftarrow Q.front$;
13     $\widehat{S} \leftarrow \widehat{S} \cup \widehat{e}$;
14   **for** *each edge* $e_{ce} = (u, v) \in CE \setminus \widehat{CE}$ && $e_{ce} \leftrightsquigarrow \widehat{e}$ **do**
15         $\widehat{CE} \leftarrow \widehat{CE} \cup e_{ce}$; $UB_2.e_{ce} \leftarrow \mathcal{L}.UB_1(e_{ce})$;
16   push $(e_{ce}, UB_2.e_{ce})$ into Q;
17   **for** $m = 1$ *to* θ_3 **do**
18         $\widehat{e} = (\widehat{u}, \widehat{v})$;
19   **if** $SG[m][\widehat{u}] == 1$ && $SG[m][\widehat{v}] == 0$ **then**
20             $SG[m][\widehat{v}] \leftarrow 1$;
21   **for** *each* $u' \leftrightsquigarrow \widehat{e}$ *in* G^m_{sg} **do**
22                 $SG[m][u'] \leftarrow 1$;
23   **else**
24   continue;
25 **return** $\widehat{S}$

Table 2: The Description of Dataset

Dataset	Nodes	Temporal Edges	d_{avg}	Days	Type
eu-core	986	332,334	25.28	803	Directed
CollegeMsg	1,899	59,835	10.69	193	Directed
mathoverflow	21,688	107,581	4.17	2,350	Directed
ask-ubuntu	137,517	280,102	1.91	2,613	Directed
stack-overflow	2,464,606	17,823,525	6.60	2,774	Directed

Table 3: Parameters and their values

Parameter	Values	Default
Q	[20, 40, 60, 80, 100]	80
$\|\mathcal{U}\|$	[1, 2, 4, 6, 8]	6
l	[1, 10, 20, 40] or [1, 2, 3, 4]	10 or 2
T	[20, 40, 60, 80, 100]	100

number of candidate edges probing for influence spread estimation, which is the bottleneck of the SBG algorithm.

6 EXPERIMENTAL EVALUATION

In this section, we present the experimental evaluation of our proposed approaches for the RTlL queries: the sketch based greedy algorithm (**SBG**) in Section 4.2; the candidate edges pruning method to accelerate SBG (**CE-SBG**) in Section 4.4; and the Order-based SBG solution (**O-SBG**) in Section 5.4.

6.1 Experimental Setting

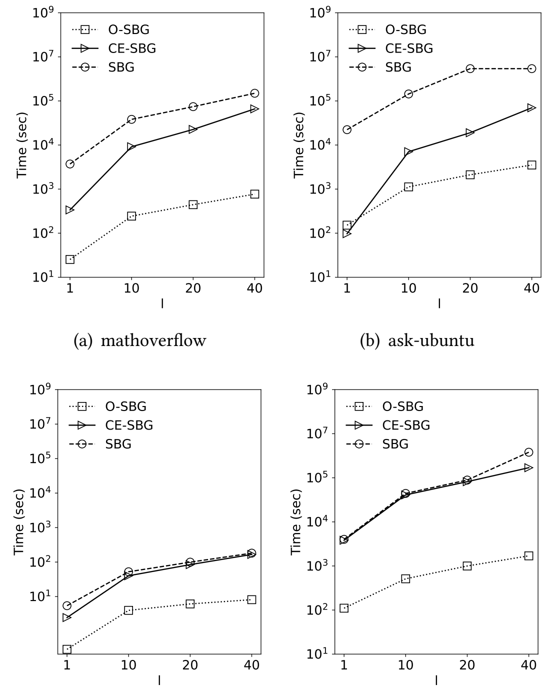
We implement the algorithms using Python 3.6 on Windows environment with 2.90GHz Intel Core i7-10700 CPU and 64GB RAM.

Baseline. To the best of our knowledge, no existing work investigates the RTlR problem. To further validate, we use our SBG algorithm as the baseline algorithm to compare with CE-SBG and O-SBG. This is because the well-known RIS based IM methods [34, 40] are hardly used in the RTlR query (*i.e., mentioned in Section 4.1*). Meanwhile, our SBG algorithm is extended from the FI-sketch IM method (*i.e., SG algorithm [10]*), while the SG algorithm performs well within the existing IM efforts, which has been validated in the state-of-the-art IM benchmark study [1].

Datasets. We conduct the experiments using five publicly available datasets from the *Large Network Dataset Collection* [2]: *eu-core*, *CollegeMsg*, *mathoverflow*, *ask-ubuntu*, and *stack-overflow*. The statistics of the datasets are shown in Table 2. We have averagely divided all datasets into T graph snapshots (*e.g.,* $G_t = (V, E_t)$, $t \in [1, T]$), where V is the node and E_t is the edges appearing in the time period of t in each dataset.

Parameter Configuration. Table 3 presents the parameter settings. We consider four parameters in our experiments: the number of queries Q, the size of given users group $|\mathcal{U}|$, reconnecting edges size l, and the number of snapshots T. Besides, the near future snapshot G_{T+1} is generated by using the recent link prediction method [50] In each experiment, if one parameter varies, we use the default values for the other parameters. Besides, we set $\theta_3 = 200$, which is consistent with [1].

[2]http://snap.stanford.edu/data/index.html



(a) mathoverflow (b) ask-ubuntu

(c) CollegeMsg (d) eu-core

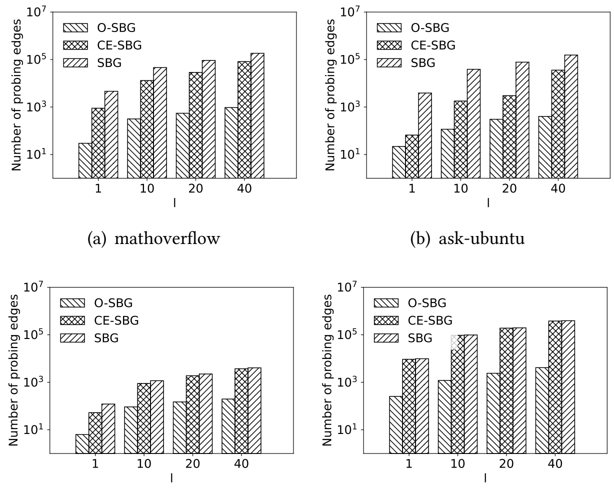
Figure 4: Time cost of algorithms with varying l

6.2 Efficiency Evaluation

We study the efficiency of the approaches for the RTlL problem regarding running time under different parameter settings.

6.2.1 Varying Reconnecting Edges Set Size l . Figure 4 shows the average running time of our proposed methods by varying l between 1 to 40. The running time of the algorithms follows similar trends, where *SBG* consumes maximum time to process an RTlR query. On average, *O-SBG* is 65 to 99 times faster than *CE-SBG*, and 90 to 167 times faster than *SBG*. Also, *CE-SBG* is about 3 to 11 times faster than *SBG* in different datasets when l varies from 2 to 40. Notably, when l is larger than 30, the *SBG* algorithm fails to return the result of the RTlR query within one day. As expected, the running time of both three approaches significantly increases when l is varied from 1 to 40. Besides, the growth of running time in *O-SBG* is much slower than the other two algorithms. This is because the probing candidate edges will increase in all three approaches when l increases, and *O-SBG* has the smallest number of probing candidate edges among the three approaches (refer to Figure 5).

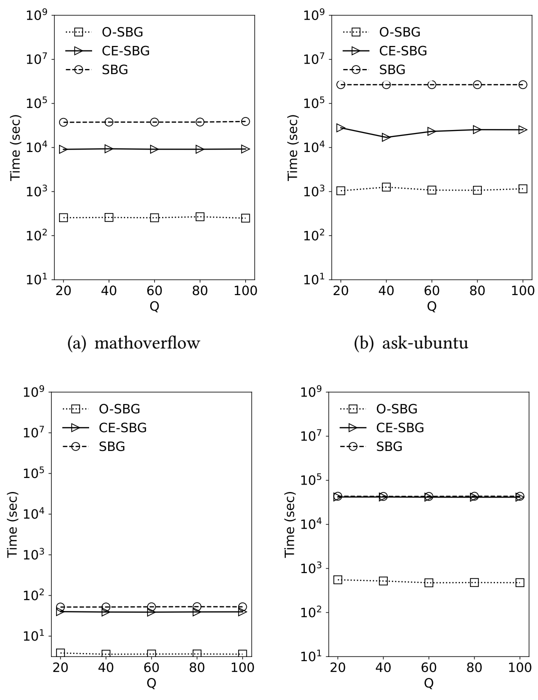
The number of probing candidate edges of *SBG*, *CE-SBG*, and *O-SBG* with varying l are presented in Figure 5(a)-5(d). As can be seen, the probing candidate edges of *O-SBG* is much less than *SBG* and *CE-SBG* for all values of l. For example, when l = 20, the probing candidate edges of *SBG*, *CE-SBG*, and *O-SBG* in *mathoverflow* are 91, 850, 28, 588, and 547, respectively. Besides, the number of probing candidate edges increases in all three approaches with the increase of l, and *O-SBG* probing the least number of candidate edges in all three approaches. This result has verified the above explanation



(a) mathoverflow (b) ask-ubuntu

(c) CollegeMsg (d) eu-core

Figure 5: Number of probing edges of algorithms



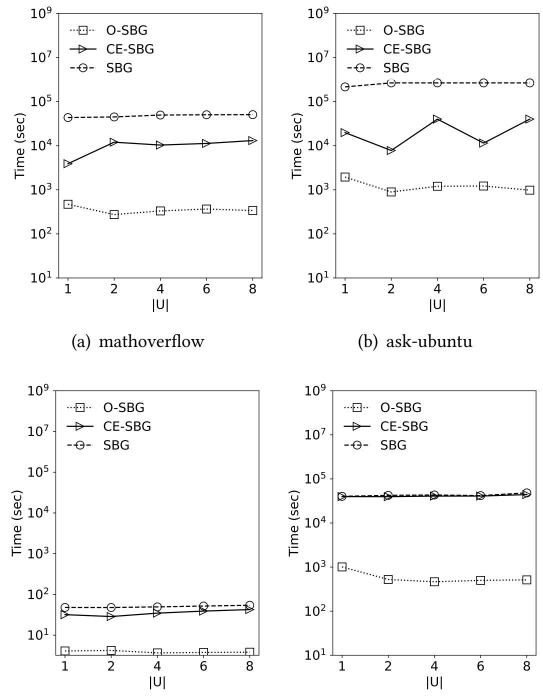
(a) mathoverflow (b) ask-ubuntu

(c) CollegeMsg (d) eu-core

Figure 6: Time cost of algorithms with varying Q

about why *O-SBG* performs better than the other two approaches with varying l.

6.2.2 Varying Number of Queries Q. We compare the performance of different approaches by varying the number of RTlR queries from 20 to 100. Figure 6 shows the average running time of *SBG*, *CE-SBG*, and *O-SBG* on the four datasets. As we can see, *O-SBG* is significantly efficient than *SBG* and *CE-SBG*. Specifically, *O-SBG* performs two to three orders of magnitude faster than *SBG* and one to two orders of magnitude faster than *CE-SBG* in all datasets, respectively.



(a) mathoverflow (b) ask-ubuntu

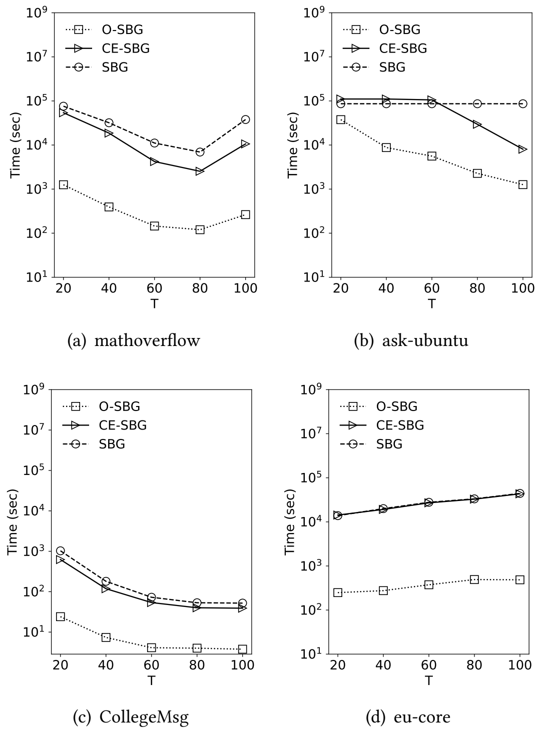
(c) CollegeMsg (d) eu-core

Figure 7: Time cost of algorithms with varying $|\mathcal{U}|$

6.2.3 Varying Users Group Size $|\mathcal{U}|$. Figure 7 shows the running time of the approaches by varying the size of users group $\mathcal{U}$ from 1 to 8. The results show similar findings that *O-SBG* outperforms *CE-SBG* and *SBG* as it utilizes the two step bounds to significantly reduce the probing candidate edges. For example, *O-SBG* can reduce the running time by around 150 times and 31 times compared with *SBG* and *CE-SBG* respectively under different $|\mathcal{U}|$ settings on the *mathoverflow* dataset.

6.2.4 Varying Snapshot Size T. We compare the efficiency of our proposed algorithms by varying the graph snapshots size T from 20 to 100. Figure 8 presents the running time with varied values of T. The results show similar finding that *O-SBG* outperforms *SBG* and *CE-SBG* in all datasets. Besides, we notice a similar running time trend in the proposed three methods when T varies. Note that the running time does not always keep the same correlation with the varies of T. This is because the performance of all three proposed approaches highly depends on the graph structure, and the number of snapshots does not show a perceptible effect on the network structure.

6.2.5 Performance in the Hyper Scale Networks. We further study the performance of different approaches on *mathoverflow*, which is a huge dataset with 2, 464, 606 nodes and 17, 823, 525 edges. It is noticed that *SBG* and *CE-SBG* cannot get results in a valid time period on *mathoverflow*, while *O-SBG* can get the results in a valid



(a) mathoverflow (b) ask-ubuntu

(c) CollegeMsg (d) eu-core

Figure 8: Time cost of algorithms with varying T

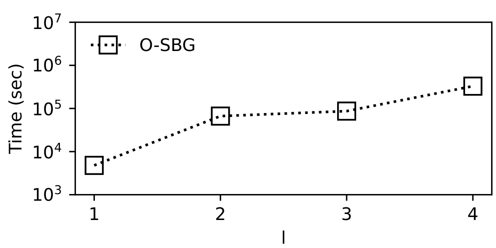


Figure 9: Performance of O-SBG on stack-overflow

period by varying l from 1 to 4. Figure 9 reports the average running time of *O-SBG* on *mathoverflow*. As we can see, the running time of *O-SBG* scales linearly with the increase of l.

6.3 Effectiveness Evaluation

In this experiment, we evaluate the number of expanding influence users produced by the RTlL problem with different datasets and approaches in Figure 10 - Figure 12 by varying one parameter and setting the others as defaults. As can be seen, the average number of influenced users of RTlR queries in dense graphs is significantly larger than in sparse graphs for all three approaches. Figure 10 shows the average number of influenced users of all three approaches *O-SBG*, *CE-SBG*, and *SBG* on four datasets with varying Q. For example, in Figure 10(a), *O-SBG*, *CE-SBG*, and *SBG* algorithms return back 39, 23, 20 number of influenced users on average when $Q = 20$ in *mathoverflow* (*i.e., nodes* $= 21,688$, *temporal edges* $= 107,581$, *average degree* $= 4.96$), respectively. Meanwhile, in Figure 10(d), *O-SBG*, *CE-SBG*, and *SBG* algorithms return back 102, 165, 164 number of influenced users on average when $Q = 20$ in *eu-core* (*i.e., nodes* $= 986$, *temporal edges* $= 332,334$, *average degree* $= 25.28$), respectively. Similar pattern can also be found in Figure 11 - Figure 12 as more influenced users be returned in dense graphs than in sparse graphs. In addition, Figure 11 reports that the influenced users of all three approaches do not always keep the same correlation with the increases of $\mathcal{U}$. Figure 12 shows that the number of influenced users by all three approaches significantly increases when l changes from 1 to 40. For example, the numbers of influenced users by *O-SBG*, *CE-SBG*, and *SBG* when setting l as 40 are 23 times, 11 times, and 15 times larger than setting l as 1 in the *mathoverflow* dataset. From the above experimental results, we can conclude that reconnecting the top-l relationship query is necessary to maximize the benefits of expanding the influenced users of a given group.

7 RELATED WORK

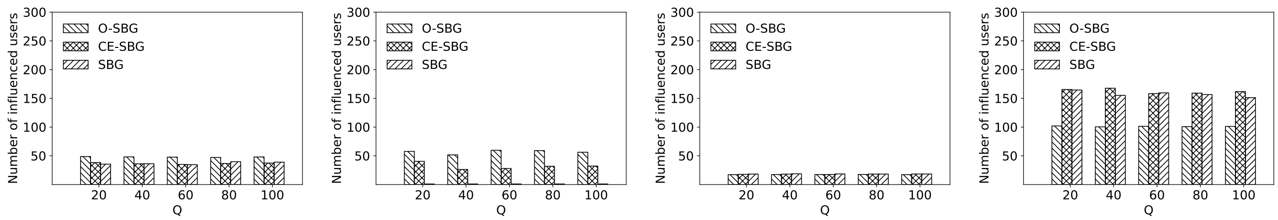
7.1 Influence Maximization

Influence maximization (IM) was first formulated by Domingos et al. [13] as an algorithmic problem in probabilistic methods. Later on, Kempe et al. [23] modeled IM as an algorithmic problem in 2003. As the IM problem is NP-hard, all existing methods focus on approximate solutions, and a keystone of these algorithmic IM studies is the greedy framework. The existing IM algorithms can be categorized into three categories: *simulation-based*, *proxy-based*, and *sketch-based.*

Simulation-based approaches. The key idea of these approaches is to estimate the influence spread $I(S)$ of given users set S by using the *Monte Carlo* (MC) simulations of the diffusion process [23, 27, 54]. Specifically, for a given users set S, the simulation-based approaches simulate the randomized diffusion process with S for R times. Each time they count the number of active users after the diffusion ends, and then take the average of these counts over the R times. The accuracy of these approaches is positively associated with the number of R. The simulation-based approaches have the advantage of diffusion model generality, and these approaches can be incorporated into any classical influence diffusion model. However, the time complexity of these approaches are cost-prohibitive, which would hardly be used for dealing with sizeable networks.

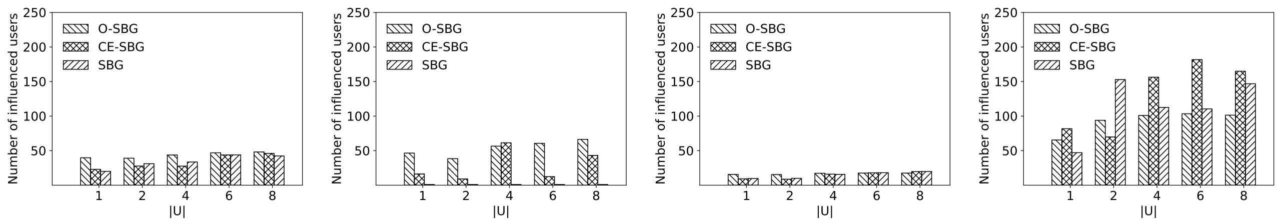
Proxy-based approaches. Instead of running heavy MC simulation, the proxy-based approaches estimate the influence spread of given users by using the proxy models. Intuitively, there are two branches of the proxy-based approaches, including (1) Estimate the influence spread of given users by transforming it to easier problems (*e.g., Degree and PageRank*) [6, 15]; and (2) Simplify the typical diffusion model (*e.g., IC model*) to a deterministic model (*e.g., MIA model*) [6] or restrict the influence propagation range of given users under the typical diffusion model to the local subgraph [16], to precisely compute the influence spread of given users. Compared with the simulation-based approach, a proxy-based approach offers significant performance improvements but lacks theoretical guarantees.

Sketch-based approaches. To avoid running heavy MC simulations and reserve the theoretical guarantee, the sketch-based approaches [3, 9, 10, 33, 35, 40] pre-compute a number of sketches



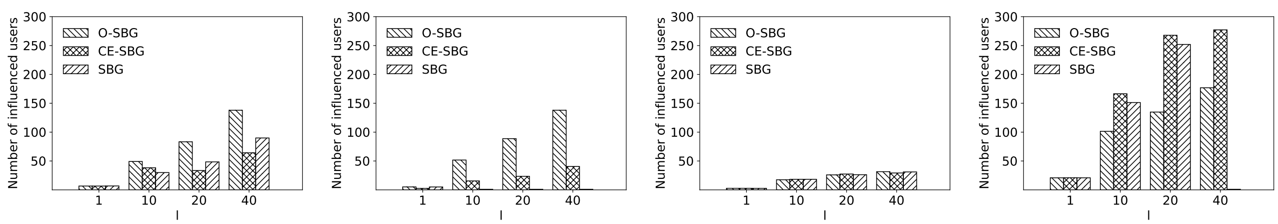
(a) mathoverflow (b) ask-ubuntu (c) CollegeMsg (d) eu-core

Figure 10: Number of influenced users with varying Q



(a) mathoverflow (b) ask-ubuntu (c) CollegeMsg (d) eu-core

Figure 11: Number of influenced users with varying $|\mathcal{U}|$



(a) mathoverflow (b) ask-ubuntu (c) CollegeMsg (d) eu-core

Figure 12: Number of influenced users with varying l

under a specific diffusion model, and then speed up the influence evaluation based on the constructed sketches. Compared with the simulation-based approaches, the sketch-based approaches have a lower time complexity under a theoretical guarantee. Unfortunately, the sketch-based approaches are not generic to all diffusion models because the generated sketches of the sketch-based approaches are relay on the underlying diffusion models.

7.2 Link Prediction

Link prediction (LP) is an important network-related problem, first proposed by Liben-Nowell et al. [31]. The LP problem aims to infer the existence of new links or still unknown interactions between pairs of nodes based on the currently observed links. After decades study, a series of LP methods were proposed, including: similarity approaches [18, 55], probabilistic approaches [12, 44], hybrid approaches [46, 52], and deep learning approaches [38, 49, 50].

In this paper, we use the SEAL method [49, 50] to predict the structure of the near future (*i.e., time point t*) snapshot graph (*i.e.,* G_t) for a given evolving graph. Furthermore, for each given users group $\mathcal{U}$, our RT*l*R query problem aims to reconnect a set of edges in G_t to maximize the number of influenced users of $\mathcal{U}$ in G_t, which is quite distinct from all existing IM works.

8 CONCLUSION

In this paper, we studied the problem of *Reconnecting Top-l Relationships* (RT*l*R), which aims to find l previous existing relationships but being estranged subsequently, such that reconnecting these relationships would maximize the influence spread of given users group. We have shown that the RT*l*L query problem is NP-hard. We developed a FI-Sketch based greedy (SBG) algorithm to solve this problem. We further devised an edge reducing method to prune the candidate edges that the given users' group cannot reach. Moreover, an order-based SBG method has been designed by utilizing the submodular characteristic of the RT*l*L query and two well-designed upper bounds. Lastly, the extensive performance evaluations on real datasets also revealed the practical efficiency and effectiveness of our proposed method. In the future, we will focus on developing more efficient approaches to deal with the RT*l*R queries in hyper scale networks.

REFERENCES

[1] Akhil Arora, Sainyam Galhotra, and Sayan Ranu. 2017. Debunking the myths of influence maximization: An in-depth benchmarking study. In *SIGMOD*. 651–666.

[2] Ruben Becker, Federico Corò, Gianlorenzo D'Angelo, and Hugo Gilbert. 2020. Balancing spreads of influence in a social network. In *AAAI*. 3–10.

[3] Christian Borgs, Michael Brautbar, Jennifer T. Chayes, and Brendan Lucier. 2014. Maximizing Social Influence in Nearly Optimal Time. In *SODA*. 946–957.

[4] Jacqueline Johnson Brown and Peter H Reingen. 1987. Social ties and word-of-mouth referral behavior. *Journal of Consumer research* 14, 3 (1987), 350–362.
[5] Taotao Cai, Jianxin Li, Ajmal Mian, Rong-Hua Li, Timos Sellis, and Jeffrey Xu Yu. 2022. Target-Aware Holistic Influence Maximization in Spatial Social Networks. *IEEE Trans. Knowl. Data Eng.* 34, 4 (2022), 1993–2007.
[6] Wei Chen, Chi Wang, and Yajun Wang. 2010. Scalable influence maximization for prevalent viral marketing in large-scale social networks. In *SIGKDD*. 1029–1038.
[7] Wei Chen, Weizhong Zhang, and Haoyu Zhao. 2020. Gradient Method for Continuous Influence Maximization with Budget-Saving Considerations. In *AAAI*. 43–50.
[8] Xiaodong Chen, Guojie Song, Xinran He, and Kunqing Xie. 2015. On Influential Nodes Tracking in Dynamic Social Networks. In *SIAM*, Suresh Venkatasubramanian and Jieping Ye (Eds.). 613–621.
[9] Suqi Cheng, Huawei Shen, Junming Huang, Guoqing Zhang, and Xueqi Cheng. 2013. StaticGreedy: Solving the Scalability-Accuracy Dilemma in Influence Maximization. In *CIKM*. 509–518.
[10] Edith Cohen, Daniel Delling, Thomas Pajor, and Renato F. Werneck. 2014. Sketch-Based Influence Maximization and Computation: Scaling up with Guarantees. In *CIKM*. 629–638.
[11] Apurba Das, Michael Svendsen, and Srikanta Tirthapura. 2019. Incremental maintenance of maximal cliques in a dynamic graph. *The VLDB Journal* 28, 3 (2019), 351–375.
[12] Sima Das and Sajal K Das. 2017. A probabilistic link prediction model in time-varying social networks. In *2017 IEEE International Conference on Communications (ICC)*. 1–6.
[13] Pedro Domingos and Matt Richardson. 2001. Mining the network value of customers. In *SIGMOD*. 57–66.
[14] Shanshan Feng, Xuefeng Chen, Gao Cong, Yifeng Zeng, Yeow Meng Chee, and Yanping Xiang. 2014. Influence Maximization with Novelty Decay in Social Networks. In *AAAI*. 37–43.
[15] Sainyam Galhotra, Akhil Arora, and Shourya Roy. 2016. Holistic influence maximization: Combining scalability and efficiency with opinion-aware models. In *ICDM*. 743–758.
[16] Amit Goyal, Wei Lu, and Laks VS Lakshmanan. 2011. Simpath: An efficient algorithm for influence maximization under the linear threshold model. In *ICDM*. 211–220.
[17] Jing Guo, Peng Zhang, Chuan Zhou, Yanan Cao, and Li Guo. 2013. Personalized influence maximization on social networks. In *CIKM*. 199–208.
[18] Yu-lin He, James NK Liu, Yan-xing Hu, and Xi-zhao Wang. 2015. OWA operator based link prediction ensemble for social network. *Expert Systems with Applications* 42, 1 (2015), 21–50.
[19] Wassily Hoeffding. 1963. Probability Inequalities for Sums of Bounded Random Variables. *J. Amer. Statist. Assoc.* 58, 301 (1963), 13–30.
[20] Shixun Huang, Zhifeng Bao, J Shane Culpepper, and Bang Zhang. 2019. Finding temporal influential users over evolving social networks. In *ICDE*. 398–409.
[21] Xiaowei Jia, Xiaoyi Li, Nan Du, Yuan Zhang, Vishrawas Gopalakrishnan, Guangxu Xun, and Aidong Zhang. 2021. Tracking Community Consistency in Dynamic Networks: An Influence-Based Approach. *IEEE Trans. Knowl. Data Eng.* 33, 2 (2021), 782–795.
[22] Richard M Karp. 1972. Reducibility among combinatorial problems. In *Complexity of Computer Computations*. 85–103.
[23] David Kempe, Jon M. Kleinberg, and Éva Tardos. 2003. Maximizing the spread of influence through a social network. In *SIGKDD*. 137–146.
[24] David Kempe, Jon M. Kleinberg, and Éva Tardos. 2015. Maximizing the Spread of Influence through a Social Network. *Theory Comput.* 11 (2015), 105–147.
[25] Dirk P Kroese, Tim Brereton, Thomas Taimre, and Zdravko I Botev. 2014. Why the Monte Carlo method is so important today. *Wiley Interdiscip Rev Comput Stat* 6, 6 (2014), 386–392.
[26] Jure Leskovec, Lars Backstrom, Ravi Kumar, and Andrew Tomkins. 2008. Microscopic evolution of social networks. In *SIGKDD*. 462–470.
[27] Jure Leskovec, Andreas Krause, Carlos Guestrin, Christos Faloutsos, Jeanne VanBriesen, and Natalie Glance. 2007. Cost-effective outbreak detection in networks. In *SIGKDD*. 420–429.
[28] Jianxin Li, Taotao Cai, Ke Deng, Xinjue Wang, Timos Sellis, and Feng Xia. 2020. Community-diversified influence maximization in social networks. *Inf. Syst.* 92 (2020), 101522.
[29] Yuchen Li, Ju Fan, Dongxiang Zhang, and Kian-Lee Tan. 2017. Discovering your selling points: Personalized social influential tags exploration. In *ICDM*. 619–634.
[30] Yuchen Li, Dongxiang Zhang, and Kian-Lee Tan. 2015. Real-time Targeted Influence Maximization for Online Advertisements. In *PVLDB*. 1070–1081.
[31] David Liben-Nowell and Jon M. Kleinberg. 2003. The link prediction problem for social networks. In *CIKM*. 556–559.
[32] Wei Lu, Wei Chen, and Laks VS Lakshmanan. 2015. From competition to complementarity: comparative influence diffusion and maximization. *arXiv preprint arXiv:1507.00317* (2015).
[33] Hung T Nguyen, Tri P Nguyen, NhatHai Phan, and Thang N Dinh. 2017. Importance sketching of influence dynamics in billion-scale networks. In *ICDM*. 337–346.
[34] Hung T Nguyen, My T Thai, and Thang N Dinh. 2016. Stop-and-stare: Optimal sampling algorithms for viral marketing in billion-scale networks. In *SIGMOD*. 695–710.
[35] Naoto Ohsaka, Takuya Akiba, Yuichi Yoshida, and Ken-ichi Kawarabayashi. 2014. Fast and accurate influence maximization on large networks with pruned monte-carlo simulations. In *AAAI*.
[36] Han-Ching Ou, Chung-Kuang Chou, and Ming-Syan Chen. 2016. Influence maximization for complementary goods: Why parties fail to cooperate?. In *CIKM*. 1713–1722.
[37] Jiamin Ou, Vincent Buskens, Arnout van de Rijt, and Debabrata Panja. 2022. Influence maximization under limited network information: Seeding high-degree neighbors. *CoRR* abs/2202.03893 (2022).
[38] Mahmudur Rahman, Tanay Kumar Saha, Mohammad Al Hasan, Kevin S Xu, and Chandan K Reddy. 2018. Dylink2vec: Effective feature representation for link prediction in dynamic networks. *arXiv preprint arXiv:1804.05755* (2018).
[39] Ashwini Kumar Singh and Lakshmanan Kailasam. 2021. Link prediction-based influence maximization in online social networks. *Neurocomputing* 453 (2021), 151–163.
[40] Youze Tang, Xiaokui Xiao, and Yanchen Shi. 2014. Influence maximization: Near-optimal time complexity meets practical efficiency. In *SIGMOD*. 75–86.
[41] Youze Tang, Xiaokui Xiao, and Yanchen Shi. 2014. Influence maximization: near-optimal time complexity meets practical efficiency. In *SIGMOD*. 75–86.
[42] Alan Tsang, Bryan Wilder, Eric Rice, Milind Tambe, and Yair Zick. 2019. Group-Fairness in Influence Maximization. In *IJCAI*. 5997–6005.
[43] Dimitris Tsaras, George Trimponias, Lefteris Ntaflos, and Dimitris Papadias. 2021. Collective Influence Maximization for Multiple Competing Products with an Awareness-to-Influence Model. *Proc. VLDB Endow.* 14, 7 (2021), 1124–1136.
[44] Tong Wang, Xing-Sheng He, Ming-Yang Zhou, and Zhong-Qian Fu. 2017. Link prediction in evolving networks based on popularity of nodes. *Scientific Reports* 7, 1 (2017), 1–10.
[45] Yu Wang, Gao Cong, Guojie Song, and Kunqing Xie. 2010. Community-based greedy algorithm for mining top-K influential nodes in mobile social networks. In *SIGKDD*. 1039–1048.
[46] Zhiqiang Wang, Jiye Liang, and Ru Li. 2018. A fusion probability matrix factorization framework for link prediction. *Knowledge-Based Systems* 159 (2018), 72–85.
[47] Miao Xie, Qiusong Yang, Qing Wang, Gao Cong, and Gerard De Melo. 2015. Dynadiffuse: A dynamic diffusion model for continuous time constrained influence maximization. In *AAAI*. 346–352.
[48] Amulya Yadav, Bryan Wilder, Eric Rice, Robin Petering, Jaih Craddock, Amanda Yoshioka-Maxwell, Mary Hemler, Laura Onasch-Vera, Milind Tambe, and Darlene Woo. 2018. Bridging the gap between theory and practice in influence maximization: Raising awareness about HIV among homeless youth.. In *IJCAI*. 5399–5403.
[49] Muhan Zhang and Yixin Chen. 2018. Link prediction based on graph neural networks. *Advances in Neural Information Processing Systems* 31 (2018), 5165–5175.
[50] Muhan Zhang, Pan Li, Yinglong Xia, Kai Wang, and Long Jin. 2021. Labeling Trick: A Theory of Using Graph Neural Networks for Multi-Node Representation Learning. *Advances in Neural Information Processing Systems* 34 (2021).
[51] Ping Zhang, Zhifeng Bao, Yuchen Li, Guoliang Li, Yipeng Zhang, and Zhiyong Peng. 2020. Towards an Optimal Outdoor Advertising Placement: When a Budget Constraint Meets Moving Trajectories. *ACM Trans. Knowl. Discov. Data* 14, 5 (2020), 1–32.
[52] Qi Zhang, Tingting Tong, and Shunyao Wu. 2020. Hybrid link prediction via model averaging. *Physica A: Statistical Mechanics and its Applications* 556 (2020), 124772.
[53] Yipeng Zhang, Yuchen Li, Zhifeng Bao, Baihua Zheng, and HV Jagadish. 2021. Minimizing the Regret of an Influence Provider. In *SIGMOD*. 2115–2127.
[54] Chuan Zhou, Peng Zhang, Wenyu Zang, and Li Guo. 2015. On the upper bounds of spread for greedy algorithms in social network influence maximization. *IEEE Trans. Knowl. Data. Eng.* 27, 10 (2015), 2770–2783.
[55] Tao Zhou, Yan-Li Lee, and Guannan Wang. 2021. Experimental analyses on 2-hop-based and 3-hop-based link prediction algorithms. *Physica A: Statistical Mechanics and Its Applications* 564 (2021), 125532.
[56] Linhong Zhu, Dong Guo, Junming Yin, Greg Ver Steeg, and Aram Galstyan. 2016. Scalable Temporal Latent Space Inference for Link Prediction in Dynamic Social Networks. *IEEE Trans. Knowl. Data. Eng.* 28, 10 (2016), 2765–2777.