Hyperparametric Robust and Dynamic Influence Maximization

Arkaprava Saha[1,2], Bogdan Cautis[3], Xiaokui Xiao[4], Laks V.S. Lakshmanan[5]
[1]DesCartes Program, CNRS@CREATE, Singapore
[2]Laboratoire d'Informatique de Grenoble, Université Grenoble Alpes, Grenoble, Auvergne-Rhône-Alpes, France
[3]Laboratoire Interdisciplinaire des Sciences du Numérique, Université Paris-Saclay, Paris, Île-de-France, France
[4]School of Computing, National University of Singapore, Singapore
[5]Department of Computer Science, University of British Columbia, Vancouver, British Columbia, Canada
arkaprava.saha@{cnrsatcreate.sg, univ-grenoble-alpes.fr}, bogdan.cautis@universite-paris-saclay.fr, xkxiao@nus.edu.sg, laks@cs.ubc.ca

Abstract

We study the problem of robust influence maximization in dynamic diffusion networks. In line with recent works, we consider the scenario where the network can undergo insertion and removal of nodes and edges, in discrete time steps, and the influence weights are determined by the features of the corresponding nodes and a global hyperparameter. Given this, our goal is to find, at every time step, the seed set maximizing the worst-case influence spread across all possible values of the hyperparameter. We propose an approximate solution using multiplicative weight updates and a greedy algorithm, with provable quality guarantees. Our experiments validate the effectiveness and efficiency of the proposed methods.

Code and Datasets — https://github.com/ArkaSaha/RDIM
Extended version — https://arxiv.org/abs/2412.11827

1 Introduction

Motivated by applications in viral marketing (Domingos and Richardson 2001), infection detection (Leskovec et al. 2007), misinformation mitigation (Simpson, Hashemi, and Lakshmanan 2022), information and influence diffusion over social networks have been extensively studied (Gionis, Terzi, and Tsaparas 2013; Tang, Shi, and Xiao 2015; Li et al. 2018). The interest in and the applicability of studies of information propagation goes beyond social media, as diffusion phenomena and algorithms for optimizing over viral mechanisms may fit in diverse domains such as Internet virus spreading (Pastor-Satorras and Vespignani 2001), biological systems (Borrvall, Ebenman, and Jonsson 2000), failures in power grids (Kinney et al. 2005), or synchronization in ad-hoc communication networks (Yu et al. 2012).

There are many ways to model information propagation in diffusion networks. A popular model is Independent Cascades (IC) (Kempe, Kleinberg, and Tardos 2003), which is *markovian* (memoryless), in that the *activation probability* of a node only depends on the current state of the system, and *local*, i.e., the activation of one node can only be caused by its neighbors. Given such a stochastic diffusion model, the classic influence maximization (IM) problem aims to find the top k seed nodes maximizing the expected number of influenced nodes in the network. The IC model has been widely and successfully used in many IM settings, effectively capturing the key aspects of diffusion phenomena.

However, the most widely used diffusion models, including IC, have two major shortcomings that limit their practical applicability. Firstly, they assume that the influence weight (or probability) between each pair of nodes is *known accurately beforehand*. When this assumption does not hold, i.e., the model is uncertain, existing IM algorithms can easily fail: small perturbations in these weights can lead to drastic changes in the solution quality (Goyal, Bonchi, and Lakshmanan 2011; Adiga et al. 2014), and even approximating the stability of a solution against small perturbations is intractable (He and Kempe 2014). However, model uncertainty is unavoidable in practice, since diffusion probabilities need to be learned from available diffusion cascades. To address this, in recent years *robust IM* has been studied (Chen et al. 2016; He and Kempe 2016; Ohsaka and Yoshida 2017; Anari et al. 2019), where the goal is to maximize the influence spread over the entire set of possible model instances. A particular line of research is where one assumes that each node in the diffusion graph has some features encoding information about it (e.g., age or follower count in a social network). Then, the influence weight between a pair of nodes is a function of their features and a global low-dimensional *hyperparameter* (Vaswani et al. 2017; Kalimeris et al. 2018). This *hyperparametric model* is intuitive (user characteristics can play an important role in users influencing each other (Cialdini 2001; Platow et al. 2005)) and computationally light when inferring probabilities, due to a smaller search space (given node features, once we find the right hyperparameter value, all probabilities are known). The inherent uncertainty in model instances arises from the potential values of the hyperparameter, which may not be known exactly, but only within a certain range.

Secondly, most existing IM algorithms are designed for a *static network*. Yet, in practice, networks are rarely static. For example, in microblogging (e.g., X a.k.a. Twitter), users may join and leave, may also create connections or remove them, at any time. Similarly, in ad-hoc communication networks between mobile devices (e.g., drones), which must perform synchronization tasks regularly, communication links get made or broken as the nodes move about in an urban landscape. In such scenarios, an IM solution may be required frequently and must be readily available at any

given moment. Recent works (Ohsaka et al. 2016; Wang et al. 2017; Peng 2021) studied *dynamic IM*, i.e., they aim to maintain a seed set with high influence spread over time, with just incremental computation per update to the network. Among these, (Peng 2021) is the only one to (i) propose a solution whose amortized running time matches that of the SOTA offline IM algorithms (with a poly-logarithmic overhead in the number of nodes) and (ii) provide approximation guarantees on the quality of the returned seed set.

We address both shortcomings of existing IM algorithms by studying the problem of *robust influence maximization in dynamic diffusion networks*. To our knowledge, we are the first to study this problem, featuring the combination of robustness and dynamicity. A practical challenge in learning any such diffusion model from available cascades is that available data may be limited. Thus, the learning algorithm should have a low sample complexity. We adopt the hyperparametric IC model of (Kalimeris, Kaplun, and Singer 2019), which enables us to meet this requirement. Specifically, we aim to answer the following question: *Under the hyperparametric model, is there a solution to robust IM over dynamic diffusion networks which, at every time step, can return a seed set with quality guarantees while being significantly more efficient than running IM from scratch?* To this end, we design an approach, called **R**obust **I**nfluence **M**aximization over **E**volving networks (**RIME**), which runs multiplicative weight updates (MWU) (Arora, Hazan, and Kale 2012; Chen et al. 2017) on values sampled from the hyperparameter space. For each weight combination, it computes the required seed set by running a greedy algorithm. We give theoretical quality guarantees for this solution and empirically show that RIME returns a good quality seed set much more efficiently than running IM from scratch upon every update to the diffusion network.

Potential Application Scenarios. We highlight three concrete applications where both uncertainty and dynamicity are prevalent. (1) In preventative health intervention (Wilder et al. 2018), algorithmic techniques are crucial to optimize targeting and delivery within a dynamic diffusion medium that blends virtual social links and real-world relationships. (2) For combating misinformation on social media (Budak, Agrawal, and El Abbadi 2011; Lakshmanan 2022) through targeted interventions, we need to quickly identify seed sets that are effective. The scale of social networks presents an added challenge, which can be addressed by leveraging sparse network representation (Mathioudakis et al. 2011) which offers significant speedup in reaching large user groups while effectively mitigating the spread of misinformation. (3) In synchronizing an ad-hoc communication network over a fleet of drones, links are inherently dynamic (more details in the extended version).

Organization & Contributions. We review classic IM (§3.1) and discuss its robust (§3.2) and dynamic (§3.3) variants. We formally state our problem, along with some basic properties and results (§3.4). In §4, we propose an approximate solution. For the *incremental* case, where nodes and edges can only be added to the network, our algorithm (§4.1) is a fusion of the MWU method HIRO from (Kalimeris, Kaplun, and Singer 2019) with (Peng 2021)'s greedy algorithm. Combining these algorithms is highly non-trivial and is one of our key contributions. RIME enjoys provable approximation guarantees (§4.2) going beyond mere extensions of those in (Kalimeris, Kaplun, and Singer 2019) and (Peng 2021). We extend RIME for the *general* case, where nodes / edges can also be removed, and establish the first known theoretical approximation guarantees for this case.

We empirically evaluate RIME, comparing with the baseline which runs from scratch after every network update. On both synthetic and real-world networks, it outperforms the baseline w.r.t. running time by several orders of magnitude, while returning a solution of comparable quality.

To sum up, we are the first to tackle the combination of robustness and dynamicity in IM, introducing RIME, the first algorithm for *robust dynamic IM*. RIME is adaptable and flexible in handling these two crucial aspects. It can handle any degree of model uncertainty, from complete knowledge to scenarios where the hyperparameter lies within a d-cube of a given size. In the absence of uncertainty, it seamlessly aligns with the incremental update algorithm from (Peng 2021), while surpassing its capabilities in fully dynamic settings. Furthermore, when dynamicity is not a concern, it gracefully reduces to the HIRO method for robust IM.

2 Related Work

Influence Maximization (IM). (Domingos and Richardson 2001) was the first work on IM, with (Kempe, Kleinberg, and Tardos 2003) later elegantly formulating the problem along with diffusion models like IC, while also proving the submodularity of the influence spread function. A plethora of works have since focused on efficient IM approximations. A breakthrough in this aspect was achieved by the concept of reverse influence sampling, introduced by (Borgs et al. 2014) and later made practical in (Tang, Shi, and Xiao 2015; Nguyen, Thai, and Dinh 2016). Several works also focus on different variants of IM, like competitive IM (Bharathi, Kempe, and Salek 2007; Carnes et al. 2007; He et al. 2012; Lin and Lui 2015; Kahr et al. 2021) and opinion maximization (Gionis, Terzi, and Tsaparas 2013; Abebe et al. 2018; Saha et al. 2023). We focus on the robust and dynamic IM variants, prior works on which are discussed next.

Robust IM. Many recent works aim to devise a solution to the IM problem which is robust to noise and uncertainty in the diffusion model. (Chen et al. 2017; Anari et al. 2019) aim to find a seed set that maximizes the minimum spread across the set of possible models. However, they assume that the number of possible models is polynomial in the size of the input, which is often not true in practice. (He and Kempe 2016; Chen et al. 2016) focus on finding the seed set maximizing the robust ratio, which is the minimum ratio (across all possible models) of the influence spread of a seed set to the optimal influence spread. (Kalimeris, Kaplun, and Singer 2019) shows that the robust ratio is not a good metric to maximize, as the seed set maximizing it can have a minimum influence spread (across all possible models) a lot worse than the corresponding optimal value. Thus, it focuses on maximizing the minimum influence spread, under the hyperpara-

metric diffusion model, assuming that the network is static.

Dynamic IM. (Chen et al. 2015; Ohsaka et al. 2016; Wang et al. 2017; Aggarwal, Lin, and Yu 2012; Liu et al. 2017; Peng 2021) are among the works that focus on IM in dynamic networks. (Chen et al. 2015) proposes an upper bound interchange method with an approximation ratio of 0.5. (Ohsaka et al. 2016) achieves a $(1-1/e)$-approximation via an algorithm based on reverse influence sampling. (Wang et al. 2017), inspired by streaming submodular maximization, designs an algorithm that maintains a constant approximate solution and is efficient in practice. However, none of them has rigorous theoretical guarantees on the amortized running time. (Peng 2021) is the first work to provide such guarantees, albeit under the assumption that the influence weights are known accurately. However, it does not provide any empirical evaluation of its proposed solution.

3 Preliminaries

A diffusion network is a graph $G = (V, E)$ where V is the set of nodes and E represents the connections between them. We denote $|V|$ and $|E|$ by n and m respectively.

3.1 Diffusion Models and Influence Functions

Independent Cascade (IC) Model. To describe diffusion between nodes in a network, various models have been used in the literature. A commonly used one is IC, which describes a discrete-step stochastic process through which information spreads in a network, from a set of initially active nodes. Each node can be active or inactive and each edge $e \in E$ in the network is associated with some probability p_e. All nodes begin as inactive. At time step $t = 0$, a subset of nodes $S \subseteq V$ (the *seeds*) is chosen and becomes active. At each step $t + 1$, every node u that became active at step t samples to influence each of its non-active neighbors v, independently with probability $p_{(u,v)}$. Activations are irreversible; a diffusion ends when there are no new activations.

Influence Functions. For a graph $G = (V, E)$ and vector of edge probabilities $\mathbf{p} \in [0, 1]^m$, the influence function $\sigma_{\mathbf{p}} : 2^V \to \mathbb{R}$ measures the expected number of nodes that will become influenced in the graph G for a seed set $S \subseteq V$:

$$\sigma_{\mathbf{p}}(S) = \sum_{A \subseteq E} r_A(S) \prod_{e \in A} p_e \prod_{e \notin A} (1 - p_e)$$

where $r_A(S)$ denotes the number of nodes reachable in G from S using only edges from A. An important property of $\sigma_{\mathbf{p}}$ is that it is non-decreasing and submodular[1] for any $\mathbf{p}$.

Influence Maximization (IM). Given a network G, a probability vector $\mathbf{p}$, and a limited budget of k seed nodes, the classic IM problem aims to find the seed set

$$S^* = \underset{S \subseteq V : |S| \leq k}{\arg\max}\ \sigma_{\mathbf{p}}(S)$$

In §3.2 we discuss robust IM, where the probability vector is not fixed and instead is assumed to belong to a given set of vectors. In §3.3 we discuss dynamic IM, where the diffusion network can change with time.

[1] A set function f is non-decreasing if for any set A and $x \notin A$, $f(A \cup \{x\}) \geq f(A)$. Furthermore, f is said to be submodular if for any $A \subseteq B$ and $x \notin B$, $f(A \cup \{x\}) - f(A) \geq f(B \cup \{x\}) - f(B)$.

3.2 Robust Influence Maximization

Given a diffusion network and a *set* of possible edge probability vectors, the robust IM problem aims to find a seed set of size k that has high spread value for every possible influence function that corresponds to a possible probability vector. One way of defining the set of possible probability vectors is via a hyperparametric model (Kalimeris, Kaplun, and Singer 2019). In this model, each node is associated with a vector of features, encoding information about it. The feature vector of each edge $e = (u, v)$, denoted $x_e \in X \subseteq [-1, 1]^d$, is obtained by the concatenation of the $d/2$-dimensional feature vectors of its endpoints u and v. The diffusion probability of the edge e is then a function of its feature vector x_e and a global hyperparameter $\theta \in \Theta = [-B, B]^d$ for some constant radius $B \in \mathbb{R}^+$. In other words, there exists a function $H : \Theta \times X \to [0, 1]$ such that $p_e = H(\theta, x_e)$. In that case, the set of possible probability vectors is given by $\mathcal{H} = \left\{ \left(H(\theta, x_e)\right)_{e \in E} : \theta \in \Theta \right\}$. Given a set $\mathcal{H}$ of probability vectors, robust IM under the hyperparametric model aims to find the seed set

$$S^* = \underset{S \subseteq V : |S| \leq k}{\arg\max} \min_{\mathbf{p} \in \mathcal{H}} \sigma_{\mathbf{p}}(S)$$

In this paper, we focus on generalized linear hyperparametric models that are 1-Lipschitz with respect to the L_1 norm, following those used in the IM literature (Vaswani et al. 2017; Kalimeris et al. 2018). For such models, we can reduce robust continuous optimization over Θ to a robust discrete one over some sampled hyperparameter values. Examples of such models include linear with $H(\theta, x_e) = \theta^T x_e$, logistic with $H(\theta, x_e) = 1/[1 + \exp(-\theta^T x_e)]$, and probit with $H(\theta, x_e) = \Phi(\theta^T x_e)$, where Φ is the CDF of the standard Gaussian distribution.

3.3 Dynamic Diffusion Networks

In diffusion scenarios, networks are mostly non-static, they can change with time in a variety of ways. As in (Peng 2021), we assume that at any time step t one of the following changes may occur: (1) a node is inserted, (2) an edge is inserted, (3) an edge is removed, or (4) a node is removed. In the *incremental setting*, only the first two types of changes are allowed. In other words, at each time step, a node or an edge can only be inserted to (not removed from) the network. In a *fully dynamic setting*, all types of changes are allowed.

Let $G_t = (V_t, E_t)$ denote the diffusion network at time t. Given a probability vector $\mathbf{p}$, the dynamic IM problem aims to find, for each time step t, the seed set

$$S_t^* = \underset{S_t \subseteq V_t : |S_t| \leq k}{\arg\max}\ \sigma_{\mathbf{p}}(S_t)$$

Remark: As in previous works, we assume that time is discretized, i.e., the actual times at which network updates occur can be mapped to discrete time steps.

3.4 Problem Statement and Hardness

We are now ready to formally state our problem of robust influence maximization in dynamic social networks.

Problem 1. *For a budget k and a dynamic diffusion network whose instance at time t is $G_t = (V_t, E_t)$ and $\mathcal{H}_t =$*

Algorithm 1: RESTART

$\delta \leftarrow n_0 \left(\frac{\delta_1}{(16(2n_0)^{Tk})}\right)^{\frac{12n_0^2}{k^3}}$
$R \leftarrow k\epsilon_1^{-2}\log\left(\frac{n_0}{\delta}\right)$
for $i = 1$ **to** l **do**
  $K_i \leftarrow 0$
  repeat
    Sample RR set with hyperparameter θ_i uniformly over V_t
    $K_i \leftarrow K_i + 1$
  until Rm_0 edges are checked
  $p_i \leftarrow \frac{K_i}{n_0}$
for all $(u, v) \in V \times V$ **do**
  $I(u, v) \leftarrow \emptyset$
for $i = 1$ **to** l **do**
  for all $v \in V$ **do**
    Generate RR set $R_{v,est,i}$ with probability p_i
    Generate RR set $R_{v,cv,i}$ with probability p_i
    for all $u \in R_{v,cv,i}$ **do**
      $I(u, v) \leftarrow I(u, v) \cup \{i\}$
for all $(u, v) \in V \times V$ **do**
  for all $i \in I(u, v)$ **do**
    $E_{cv,i} \leftarrow E_{cv,i} \cup \{(u, v)\}$
  FIND-SEEDS((u, v))

Algorithm 2: FIND-SEEDS

Input: Edge (u', v')
Output: Seed set
for $j = 1$ **to** T **do**
  for $i = 1$ **to** l **do**
    $w_j[i] \propto \exp\left(-\eta \sum_{\tau=1}^{j-1} f_{cv,i}(S^{(\tau)})\right)$
  $S^{(j)} \leftarrow$ GREEDY($\langle w_j, f_{cv}\rangle, (u', v')$)
for $i = 0$ **to** $\lceil \epsilon^{-1} \log n \rceil$ **do**
  $S_i \leftarrow$ Seed set computed by GREEDY for thread i with uniform weights
return Uniform distribution over $\left\{S^{(1)}, \ldots, S^{(T)}\right\}$

Algorithm 3: GREEDY (INSERTION)

Input: Objective function F, edge (u', v')
Output: Seed set
for $i = 0$ **to** $\lceil \epsilon^{-1} \log n \rceil$ **do**
  $u'' \leftarrow u'$
  while $F(S_i \cup \{u''\}) - F(S_i) \geq [(1+\epsilon_1)^i - F(S_i)]/k$ **and** $|S_i| < k$ **do**
    $S_i \leftarrow S_i \cup \{u''\}$
    $u'' \leftarrow \arg\max_{v'' \notin S_i} F(S_i \cup \{v''\}) - F(S_i)$
return $\arg\max_i F(S_i)$

$\left\{\left(H\left(\theta, x_e\right)\right)_{e \in E_t} : \theta \in \Theta\right\}$ is the set of possible edge probability configurations, find for all time steps t the seed set

$$S_t^* = \underset{S_t \subseteq V_t : |S_t| \leq k}{\arg\max} \min_{\mathbf{p} \in \mathcal{H}_t} \sigma_{\mathbf{p},t}\left(S_t\right)$$

where $\sigma_{\mathbf{p},t}\left(S_t\right)$ is the expected influence spread of S_t in G_t under the edge probability configuration $\mathbf{p}$.

Clearly, hardness results similar to those of robust IM or dynamic IM also hold here. Specifically, at each time step, it is **NP**-hard to (1) get any approximation better than $(1 - 1/e)$ and (2) find a seed set of size k that obtains any approximation better than $\mathcal{O}(\log n)$ (Kalimeris, Kaplun, and Singer 2019). So we look for an (α, β) bi-criteria approximation, where $\alpha, \beta \in (0, 1)$; i.e. a seed set $\hat{S}_t$ s.t. $\beta|\hat{S}_t| \leq k$ and

$$\min_{\mathbf{p} \in \mathcal{H}_t} \sigma_{\mathbf{p},t}\left(\hat{S}_t\right) \geq \alpha \max_{S_t \subseteq V_t : |S_t| \leq k} \min_{\mathbf{p} \in \mathcal{H}_t} \sigma_{\mathbf{p},t}\left(S_t\right)$$

When the hyperparametric model and the set of features are clear from the context, the probability vector $\mathbf{p}$ is defined entirely by the hyperparameter θ. In such cases, for ease of understanding, we shall use $\sigma_{\theta,t}$ instead of $\sigma_{\mathbf{p},t}$.

4 Solution

In this section, we propose an approximate solution to Problem 1 for both the incremental and the fully dynamic settings. We discuss the overall algorithm in §4.1 and conduct a theoretical analysis of its quality and running time in §4.2. Omitted pseudocodes are provided in the extended version.

4.1 Algorithm

We first provide a brief overview of our method. Let n_0 and m_0 respectively denote the initial number of nodes and edges in our network. As mentioned in the last paragraph of § 3.2, we first sample l values $\theta_1, \ldots, \theta_l$ from the hyperparameter space Θ. Then, for each sampled hyperparameter, following the SOTA in IM, we sample a few reverse reachable (RR) sets and solve a maximum coverage problem on those sets, thereby computing the seed set for the initial network. After that, when a node or an edge is added to or removed from the network, we generally run some incremental computations, e.g. modifying the previously generated RR sets and hence the seed sets. However, if some conditions are satisfied after the network update (e.g., the size of the network is doubled or halved), we restart from scratch. In the next paragraphs, we explain our method in more detail.

When our method runs for the very first time, for each $i \in [l]$, it invokes Alg. 1 – RESTART where, from each node v, it generates two RR sets $R_{v,est,i}$ and $R_{v,cv,i}$ (Lines 12-17), each with probability p_i (computed in Lines 3-9). These collections of RR sets are modelled as bipartite coverage graphs $\mathcal{H}_{est,i} = (V_{L,est,i}, V_{R,est,i}, E_{est,i})$ and $\mathcal{H}_{cv,i} = (V_{L,cv,i}, V_{R,cv,i}, E_{cv,i})$, where $(u, v) \in E_{est,i}$ (resp. $E_{cv,i}$) if and only if $u \in R_{v,est,i}$ (resp. $R_{v,cv,i}$).[2] Clearly, the number of RR sets $R_{v,cv,i}$ covered by a seed set S is equal to the number of nodes in $V_{R,cv,i}$ adjacent to S in $\mathcal{H}_{cv,i}$. This number, normalized (divided) by p_i, is denoted by $f_{cv,i}(S)$ and is used to estimate the true influence spread $\sigma_{\theta_i}(S)$ of S for hyperparameter θ_i.

We now discuss how to find the seeds via dynamic maximum coverage on the $\mathcal{H}_{cv,i}$s. As shown in Lines 18-21 of

[2] We generate two bipartite graphs to decouple the correlation between estimating the influence spread of a seed set and estimating the time required to sample an RR set (Peng 2021). $\mathcal{H}_{cv,i}$ is used for the former, while $\mathcal{H}_{est,i}$ is used for the latter.

Algorithm 4: INSERT-NODE

Input: Node $u \in V$

$V \leftarrow V \cup \{u\}$, $n \leftarrow n + 1$
if $n \geq 2n_0$ **then**
  $m_0 \leftarrow m$, $n_0 \leftarrow n$
  RESTART
else
  for $i = 1$ **to** l **do**
    $V_{L,est,i} \leftarrow V_{L,est,i} \cup \{u\}$
    $V_{L,cv,i} \leftarrow V_{L,cv,i} \cup \{u\}$
    $V_{R,est,i} \leftarrow V_{R,est,i} \cup \{u\}$ with probability p_i
    $V_{R,cv,i} \leftarrow V_{R,cv,i} \cup \{u\}$ with probability p_i

Algorithm 5: INSERT-EDGE

Input: Edge $e = (u, v) \in E$

$E \leftarrow E \cup \{e\}$, $m \leftarrow m + 1$
if $m \geq 2m_0$ **then**
  $m_0 \leftarrow m$, $n_0 \leftarrow n$
  RESTART
else
  for $i = 1$ **to** l **do**
    for all $v' \in V_{R,est,i}$ **do**
      if $(v, v') \in E_{est,i}$ **then**
        Augment RR sets $R_{v',est,i}$
        if #edges scanned to update $\mathcal{H}_{est,i} \geq 16Rm_0$ **then**
          RESTART
  for all $(u', v') \in V \times V$ **do**
    $I(u', v') \leftarrow \emptyset$
  for $i = 1$ **to** l **do**
    for all $v' \in V_{R,cv,i}$ **do**
      if $(v, v') \in E_{cv,i}$ **then**
        Augment RR sets $R_{v',cv,i}$ to $R^{new}_{v',cv,i}$
        for all $u' \in R^{new}_{v',cv,i} \setminus R_{v',cv,i}$ **do**
          $I(u', v') \leftarrow I(u', v') \cup \{i\}$
        $R^{new}_{v',cv,i} \leftarrow R_{v',cv,i}$
  for all $(u', v') \in V \times V$ **do**
    for all $i \in I(u', v')$ **do**
      $E_{cv,i} \leftarrow E_{cv,i} \cup \{(u', v')\}$
    FIND-SEEDS((u', v'))

Alg. 1, whenever an edge is added to the $\mathcal{H}_{cv,i}$s (i.e., a node is added to the corresponding RR sets), the seed set also needs to be updated by calling Alg. 2 – FIND SEEDS. Along the lines of HIRO, Alg. 2 runs T iterations of multiplicative weight updates (MWU), each computing a candidate seed set after running an approximation algorithm to maximize a convex combination F of the sampled influence functions (Line 4); higher weights are given to those with poor performance in previous iterations. Similar to (Peng 2021), the approximation algorithm used in Line 4 of Alg. 2, which is shown in Alg. 3 – GREEDY (INSERTION), maintains a seed set for each of the $\lceil \epsilon_1^{-1} \log n \rceil$ threads, where thread i corresponds to the estimate $(1+\epsilon_1)^i$ of the maximum value of F. Alg. 3 adds to each thread's seed set any node with marginal gain above a threshold. After that, it returns the seed set maximizing F. In the end, after getting the candidate seed sets using MWU, Alg. 2 returns one of those sets uniformly at random, after updating each thread's seed set to be the last one computed by Alg. 3, with F having uniform weights.

After the initial run, when a node (resp. edge) is inserted to the network, Alg. 4 – INSERT NODE (resp. 5 – INSERT EDGE) is invoked. As shown in the pseudocode, the entire process restarts from scratch (invokes Alg. 1) when the number of nodes or edges is doubled. Unless the restarting condition is met, the only computation required when a node is inserted is creating some RR sets containing only that node. When an edge is inserted, however, it can enhance the information diffusion in the network, thus more updates are needed. Specifically, whenever an edge (u, v) with probability $p_{(u,v)}$ is added, each RR set containing v but not u should be augmented by adding u with probability $p_{(u,v)}$. If it is added, we continue augmenting this RR set, i.e., adding new nodes to the set by sampling incoming edges of u and repeating the same for the new ones till no further node can be reached. For RR sets in $\mathcal{H}_{est,i}$, if this process of augmentation results in the overall number of edges in E scanned while updating $\mathcal{H}_{est}$ exceeding a threshold, we restart from scratch (Alg. 1). For RR sets in $\mathcal{H}_{cv,i}$, augmentation clearly leads to new (bipartite) edges being added, which means we need to update the seed set (Alg. 2) for each such addition.

The techniques for handling node and edge removal are shown in Alg. 6 – REMOVE-NODE and 7 – REMOVE-EDGE. In both cases, a process restarts from scratch (Alg. 1) whenever the number of nodes or edges is halved. A node removal only leads to its removal from all RR sets. When an edge (u, v) is removed, however, the diffusion pathways can be affected, and thus more processing is needed. Specifically, we need to reduce each RR set containing both u and v, where u was reached after sampling incoming edges to v, by removing u and all of its descendants in the RR set. For RR sets in $\mathcal{H}_{est,i}$, if this process results in the overall number of edges in E scanned while updating $\mathcal{H}_{est}$ exceeding[3] a threshold, we restart from scratch (Alg. 1). For RR sets in $\mathcal{H}_{cv,i}$, such an RR set reduction leads to the removal of (bipartite) edges, which requires an update of the seed set. The seed set is computed by calling the same MWU-based method as for insertions (Alg. 2). However, the approximation algorithm called in Line 4 of Alg. 2 is different for edge removals. When an edge $(u', v') \in E_{cv,i}$ is to be removed, for each thread's seed set, we remove u' and add the node with the largest marginal gain (possibly u'); after that, we add each node with marginal gain above a threshold. This is shown in Alg. 8 – GREEDY (REMOVAL).

Note that the method in (Peng 2021) needs only two coverage graphs $\mathcal{H}_{est}$ and $\mathcal{H}_{cv}$, while ours requires l such pairs of graphs to account for the noise in the edge probabilities. Also, (Peng 2021) finds seeds by invoking the greedy algorithm for maximizing coverage only once for each update to $\mathcal{H}_{cv}$, whereas to account for the edge probability noise,

[3] Removing a node and all of its descendants from an RR set requires scanning the edges in the graph structure of the RR set, which leads to the increase in the number of edges in the network scanned while updating the corresponding coverage graph.

we need to do the same via T MWU iterations, and in each iteration, we need to maximize a convex combination of l coverage functions. Fortunately, we can still provide quality guarantees for the resultant method, as shown in §4.2.

4.2 Analysis

We now analyze RIME's quality and running time (proofs provided in the extended version). We first give a quality guarantee for our greedy algorithm. Consider a run of the algorithm upon addition of a node to an RR set (so a call to Alg. 2), with the input objective function F having optimal value OPT. For any seed set S and $x < |S|$, let $S_{1:x}$ be the set of the first x nodes added to S. Also, for $y \in \{x, x+1\}$, let $F_{t_y}(S_{1:x})$ be the value of $F(S_{1:x})$ at the time the y^{th} node was added to S. For completeness, define $F_{t_{k+1}}(\cdot) = F(\cdot)$.

Lemma 1. *Our greedy algorithm returns a seed set $\hat{S}$ such that $F\left(\hat{S}\right) \geq (1 - 1/e - \gamma - \epsilon_1) \cdot OPT$, where*

$$\gamma = \begin{cases} \max_{x=1}^{|\hat{S}|} \frac{F_{t_x}(\hat{S}_{1:x}) - F_{t_{x+1}}(\hat{S}_{1:x})}{OPT} \left(1 - \frac{1}{k}\right)^{|\hat{S}|-x} & \textit{increm.} \\ \sum_{x=1}^{|\hat{S}|} \frac{F_{t_x}(\hat{S}_{1:x}) - F_{t_{x+1}}(\hat{S}_{1:x})}{OPT} \left(1 - \frac{1}{k}\right)^{|\hat{S}|-x} & \textit{fully dyn.} \end{cases}$$

To check how close the above approximation guarantee is to $1 - 1/e - \epsilon_1$, the guarantee for the SOTA on IM (Tang, Xiao, and Shi 2014; Tang, Shi, and Xiao 2015), we ran simulations on 5 networks (§5) and found the maximum difference γ between the two values across all networks to be 4.7×10^{-14} (incremental) and 0.124 (fully dynamic), showing that the algorithm approximates very well in practice.

We can now provide a guarantee for the minimum influence spread across the entire hyperparameter space Θ.

Theorem 1. *Let γ be as defined in Lemma 1. Let c_p denote the maximum number of phases during which the size of the graph remains within a factor of 2 of the original. Within each such phase, let c_r denote the number of stages within which the number of edges scanned to update $\mathcal{H}_{est,i}$ for each $i \in [l]$ remains below the threshold after which it restarts from scratch. Define $S_{tc} = \cup_{i=1}^{T} S_t^{(i)}$. With probability at least $1 - \delta_1 - \delta_2$, in every time step t,*

$$\min_{\theta \in \Theta} \sigma_{\theta,t}(S_{tc}) \geq \left(1 - \frac{1}{e} - \gamma - 3\epsilon_1\right) \max_{\substack{S \subseteq V_t: \\ |S| \leq k}} \min_{\theta \in \Theta} \sigma_{\theta,t}(S) - 2\epsilon_2$$

for $l \geq d\left(\frac{2Bdmn}{\epsilon_2}\right)^d \log\left(\frac{2Bdmn}{\epsilon_2 \delta_2}\right)$ and $T \geq \frac{2 \log l}{\epsilon_2^2}$, with amortized running time $O\left(\frac{n^2 l^2 T c_p c_r}{k^2 \epsilon_1^3} \cdot \log n \cdot \log\left(\frac{n^{Tk}}{\delta_1}\right)\right)$.

Remarks: (1) Th. 1 provides minimum values of l and T needed for a desired solution quality, with higher values leading to better solutions. However, in our experiments (see the extended version), we find that we can obtain reasonably good results even with much smaller values of l and T, and the solution quality converges (remains nearly the same) beyond those values. So in practice we can run RIME for much smaller l and T (and faster), while achieving a reasonably good solution. **(2)** For the incremental setting, the values of c_p and c_r in Th. 1 are at most $O(\log n)$ (Peng 2021). Also, according to Line 3 of Alg. 3, once all threads' seed sets have k nodes each, they will not be further updated (since they can only increase in size), which means we can also avoid bookkeeping computations like updating the marginal gain of every node for subsequent seed additions. However, this cannot be done for the fully dynamic setting because, in that case, even if a thread's seed set has k nodes, an edge deletion will require replacement of one seed with another, which requires the marginal gains of each node to be up to date (Alg. 8). These factors account for the running time being longer for the fully dynamic setting than for the incremental one, as proven empirically in §5.

5 Experiments

We evaluate next RIME's effectiveness and efficiency. It was implemented in C++, run on a Linux server with 2 GHz AMD CPU, 512GB RAM. Other experiments on parameter sensitivity or efficiency / effectiveness on larger or different kinds of networks are provided in the extended version.

Networks & Hyperparameter. We generate synthetic random networks with $n_0 = 10^x$, $x \in \{2, 3, 4, 5\}$, and $m_0 = 2n_0$. We also test our methods on the real-world Weibo social network (Zhang et al. 2013). From the original dataset, we extract the largest connected component of the subgraph containing edges with at least 30 reposts between the corresponding users. The resultant network has $n_0 = 99523$ and $m_0 = 180216$. In each of our networks, we use the sigmoid function as the hyperparametric model to determine the edge probabilities, i.e., $H(\theta, x_e) = 1/[1 + \exp(-\theta^T x_e)]$ as in (Kalimeris, Kaplun, and Singer 2019).

Features. For our synthetic networks, following (Kalimeris, Kaplun, and Singer 2019), we generate 3 random features for each node, for a 6-dimensional hyperparameter for each edge. As in §3.2, the hyperparameter space is a d-cube centred at the origin. In Weibo, we use as features user attributes (followers / friends / status counts, gender, verified status). The categorical features are binarized. The resultant hyperparameter space, with dimension $d = 16$, is a d-cube centred at the value for which the likelihood of the repost cascade data (available with the dataset) is maximized (Kalimeris et al. 2018). This makes the space of possible edge probabilities more realistic than that with the hypercube centred at the origin, while not affecting our algorithms in any way.

Network updates. For each of our networks initially with n_0 nodes and m_0 edges, we generate $2m_0$ random updates of all possible kinds (node / edge insertion / removal) for both the incremental setting and the fully dynamic setting.

Methods compared. For both the incremental setting and the fully dynamic one, we compare RIME with baselines that run the following methods from scratch after each network update:(i) HIRO (resp. (ii) BASE), which runs the method in (Kalimeris, Kaplun, and Singer 2019) by generating RR sets from θ randomly chosen nodes (resp. once from every node) for each MWU iteration; and (iii) LUGreedy (Chen et al. 2016), which assumes confidence intervals for the edge probabilities, runs the greedy algorithm twice over both the lower and the upper boundaries of the intervals, and returns the better of the two above solutions assuming the lower boundaries as true probabilities.

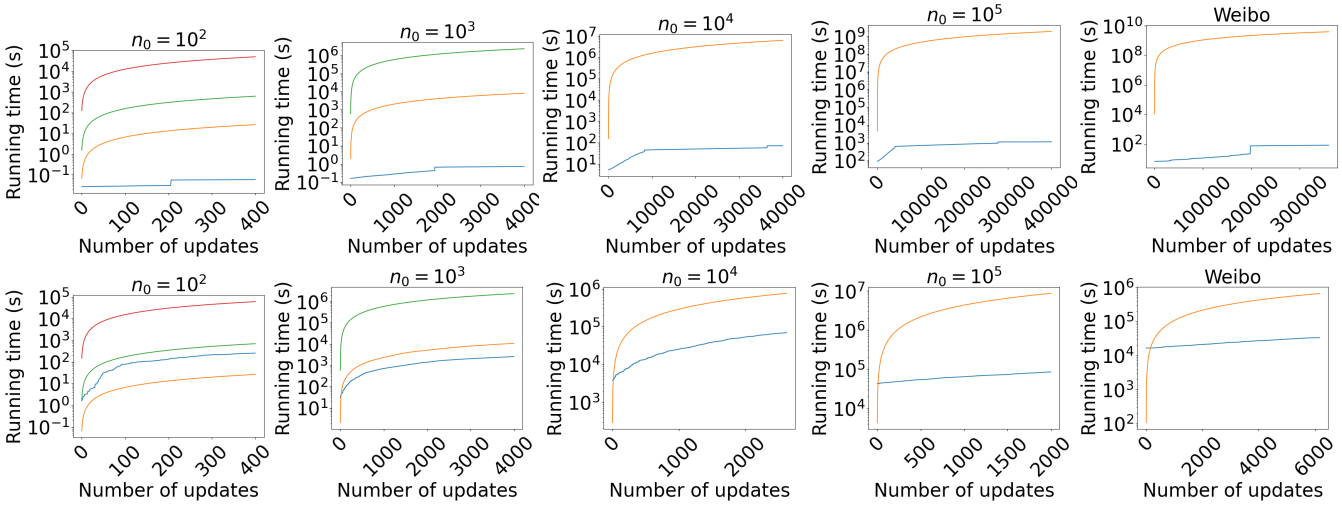

Figure 1: Running time comparison on synthetic / real-world networks for incremental (top row) / fully dynamic (bottom row) settings. The blue, yellow, green and red lines denote the methods RIME, Base, LUGreedy and HIRO respectively.

Metrics. We compare RIME's efficiency against the baselines on *total running time* (across all network updates). We compare RIME's quality by computing, after all updates, the minimum (approximate) influence spread of the returned seed sets across the sampled hyperparameter values.[4]

Efficiency. We compare the running time of RIME against the baselines in Fig. 1. A missing line means that the corresponding method did not finish in a day while running on the very first graph without any updates. It is clear that, in most cases, RIME is several orders of magnitude faster than the competitors. Also, for the incremental setting RIME is 2-3 orders of magnitude faster than that for the fully dynamic setting. This is explained by the fact that we can avoid several bookkeeping computations for the incremental setting (see the last paragraph of §4.2). Observe the jumps in the plots for RIME. They correspond to RIME invoking the algorithm restarting from scratch (Alg. 1). Notice that in the incremental setting, the slope of each line for RIME decreases after each restart. This can be explained as follows. Later restarts take place on bigger graphs than earlier ones. Thus, after later restarts, when an edge (u, v) is added, RR sets containing v are more likely to contain u as well, compared to earlier restarts, since there are more ways of reaching u on a bigger graph than on a smaller one. Note that only those RR sets containing v but not u need to be augmented (§4.1). This leads to fewer RR set augmentations after later restarts, thereby lowering the running time. Finally, we would like to point out that RIME scales much better than our baselines, at least up to graphs with 10^5 nodes.[5]

Effectiveness. The solution quality comparisons for both settings are shown in Fig. 2. A missing bar means that the corresponding method did not finish in a day while running on the initial graph without any updates. Clearly, the best baseline is HIRO, which does not scale to graphs with 10^3 nodes or more. Even for the graphs on which HIRO does complete, RIME's quality is comparable to HIRO's. LUGreedy returns lower-quality solutions than RIME, while also not scaling to graphs with 10^4 nodes or more. BASE finishes on all our graphs, but returns slightly lower-quality solutions than RIME, while being orders of magnitude slower.

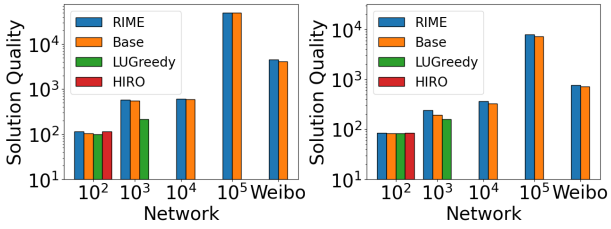

Figure 2: Solution quality comparison on synthetic / real-world networks for incremental (left) and fully dynamic (right) settings.

6 Conclusion

We studied the problem of robust influence maximization over dynamic diffusion networks, where the diffusion probabilities are a function of the node features and a global hyperparameter, and the network evolves by node / edge insertions / deletions. We proposed an approximation solution using multiplicative weight updates and a greedy algorithm, with theoretical quality guarantees. We empirically assessed our method's performance on synthetic & real networks and found that it is generally orders of magnitude faster than the baselines, while returning a seed set of comparable quality. A potential future direction is to extend the RIME framework to other diffusion models and forms of dynamicity.

[4] Each value is an average over 10 algorithm runs, generally with a small std. dev.

[5] The problem we study in this paper has several well-documented sources of hardness. Therefore, while RIME may not scale to large, Twitter-like diffusion networks, it remains reasonably applicable for networks with node counts of orders of tens-of-thousands to hundreds-of-thousands. However, we believe this scale is entirely relevant for many application scenarios, including the ones we mention in §1.

Acknowledgments

This research was supported by the National Research Foundation, Prime Minister's Office, Singapore under its Campus for Research Excellence and Technological Enterprise (CREATE) program. The computational work for this paper was performed fully on resources of the National Supercomputing Centre, Singapore (https://www.nscc.sg). The research of the last author was supported in part by a grant from the Natural Sciences and Engineering Research Council of Canada (Grant Number RGPIN-2020-05408).

References

Abebe, R.; Kleinberg, J.; Parkes, D.; and Tsourakakis, C. E. 2018. Opinion dynamics with varying susceptibility to persuasion. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 1089–1098.

Adiga, A.; Kuhlman, C. J.; Mortveit, H. S.; and Vullikanti, A. K. S. 2014. Sensitivity of diffusion dynamics to network uncertainty. *Journal of Artificial Intelligence Research*, 51: 207–226.

Aggarwal, C. C.; Lin, S.; and Yu, P. S. 2012. On influential node discovery in dynamic social networks. In *Proceedings of the 2012 SIAM International Conference on Data Mining*, 636–647.

Anari, N.; Haghtalab, N.; Naor, S.; Pokutta, S.; Singh, M.; and Torrico, A. 2019. Structured robust submodular maximization: Offline and online algorithms. In *The 22nd International Conference on Artificial Intelligence and Statistics*, 3128–3137.

Arora, S.; Hazan, E.; and Kale, S. 2012. The multiplicative weights update method: A meta-algorithm and applications. *Theory of computing*, 8(1): 121–164.

Bharathi, S.; Kempe, D.; and Salek, M. 2007. Competitive influence maximization in social networks. In *International Workshop on Web and Internet Economics*, 306–311.

Borgs, C.; Brautbar, M.; Chayes, J.; and Lucier, B. 2014. Maximizing social influence in nearly optimal time. In *Proceedings of the 25th Annual ACM-SIAM symposium on Discrete algorithms*, 946–957.

Borrvall, C.; Ebenman, B.; and Jonsson, T. 2000. Biodiversity lessens the risk of cascading extinction in model food webs. *Ecology Letters*, 3(2): 131–136.

Budak, C.; Agrawal, D.; and El Abbadi, A. 2011. Limiting the spread of misinformation in social networks. In *Proceedings of the 20th International Conference on World Wide Web*, 665–674.

Carnes, T.; Nagarajan, C.; Wild, S. M.; and van Zuylen, A. 2007. Maximizing influence in a competitive social network: A follower's perspective. In *Proceedings of the 9th International Conference on Electronic Commerce*, 351–360.

Chen, R. S.; Lucier, B.; Singer, Y.; and Syrgkanis, V. 2017. Robust optimization for non-convex objectives. In *Advances in Neural Information Processing Systems*, volume 30, 4708–4717.

Chen, W.; Lin, T.; Tan, Z.; Zhao, M.; and Zhou, X. 2016. Robust influence maximization. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 795–804.

Chen, X.; Song, G.; He, X.; and Xie, K. 2015. On influential nodes tracking in dynamic social networks. In *Proceedings of the 2015 SIAM International Conference on Data Mining*, 613–621.

Cialdini, R. 2001. Harnessing the science of persuasion. *Harvard Business Review*, 79(9): 72–79.

Domingos, P.; and Richardson, M. 2001. Mining the network value of customers. In *Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 57–66.

Gionis, A.; Terzi, E.; and Tsaparas, P. 2013. Opinion maximization in social networks. In *Proceedings of the 2013 SIAM International Conference on Data Mining*, 387–395.

Goyal, A.; Bonchi, F.; and Lakshmanan, L. V. 2011. A data-based approach to social influence maximization. *Proceedings of the VLDB Endowment*, 5(1): 73–84.

He, X.; and Kempe, D. 2014. Stability of influence maximization. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 1256–1265.

He, X.; and Kempe, D. 2016. Robust influence maximization. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 885–894.

He, X.; Song, G.; Chen, W.; and Jiang, Q. 2012. Influence blocking maximization in social networks under the competitive linear threshold model. In *Proceedings of the 2012 SIAM International Conference on Data Mining*, 463–474.

Kahr, M.; Leitner, M.; Ruthmair, M.; and Sinnl, M. 2021. Benders decomposition for competitive influence maximization in (social) networks. *Omega*, 100: 102264.

Kalimeris, D.; Kaplun, G.; and Singer, Y. 2019. Robust influence maximization for hyperparametric models. In *International Conference on Machine Learning*, 3192–3200.

Kalimeris, D.; Singer, Y.; Subbian, K.; and Weinsberg, U. 2018. Learning diffusion using hyperparameters. In *International Conference on Machine Learning*, 2420–2428.

Kempe, D.; Kleinberg, J.; and Tardos, E. 2003. Maximizing the spread of influence through a social network. In *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 137–146.

Kinney, R.; Crucitti, P.; Albert, R.; and Latora, V. 2005. Modeling cascading failures in the North American power grid. *The European Physical Journal B*, 46(1): 101–107.

Lakshmanan, L. V. S. 2022. On a quest for combating filter bubbles and misinformation. In *Proceedings of the 2022 ACM SIGMOD International Conference on Management of Data*, 2.

Leskovec, J.; Krause, A.; Guestrin, C.; Faloutsos, C.; VanBriesen, J.; and Glance, N. 2007. Cost-effective outbreak detection in networks. In *Proceedings of the 13th ACM*

SIGKDD International Conference on Knowledge Discovery and Data Mining, 420–429.

Li, Y.; Fan, J.; Wang, Y.; and Tan, K.-L. 2018. Influence maximization on social graphs: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 30(10): 1852–1872.

Lin, Y.; and Lui, J. C. 2015. Analyzing competitive influence maximization problems with partial information: An approximation algorithmic framework. *Performance Evaluation*, 91: 187–204.

Liu, X.; Liao, X.; Li, S.; Zheng, S.; Lin, B.; Zhang, J.; Shao, L.; Huang, C.; Xiao, L.; et al. 2017. On the shoulders of giants: Incremental influence maximization in evolving social networks. *Complexity*, 2017.

Mathioudakis, M.; Bonchi, F.; Castillo, C.; Gionis, A.; and Ukkonen, A. 2011. Sparsification of influence networks. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 529–537.

Nguyen, H. T.; Thai, M. T.; and Dinh, T. N. 2016. Stop-and-stare: Optimal sampling algorithms for viral marketing in billion-scale networks. In *Proceedings of the 2016 ACM SIGMOD International Conference on Management of Data*, 695–710.

Ohsaka, N.; Akiba, T.; Yoshida, Y.; and Kawarabayashi, K.-i. 2016. Dynamic influence analysis in evolving networks. *Proceedings of the VLDB Endowment*, 9(12): 1077–1088.

Ohsaka, N.; and Yoshida, Y. 2017. Portfolio optimization for influence spread. In *Proceedings of the 26th International Conference on World Wide Web*, 977–985.

Pastor-Satorras, R.; and Vespignani, A. 2001. Epidemic spreading in scale-free networks. *Physical Review Letters*, 86: 3200–3203.

Peng, B. 2021. Dynamic influence maximization. In *Advances in Neural Information Processing Systems*, volume 34, 10718–10731.

Platow, M. J.; Haslam, S. A.; Both, A.; Chew, I.; Cuddon, M.; Goharpey, N.; Maurer, J.; Rosini, S.; Tsekouras, A.; and Grace, D. M. 2005. "It's not funny if they're laughing": Self-categorization, social influence, and responses to canned laughter. *Journal of Experimental Social Psychology*, 41(5): 542–550.

Saha, A.; Ke, X.; Khan, A.; and Lakshmanan, L. V. 2023. Voting-based opinion maximization. In *IEEE 39th International Conference on Data Engineering*.

Simpson, M.; Hashemi, F.; and Lakshmanan, L. V. 2022. Misinformation mitigation under differential propagation rates and temporal penalties. *Proceedings of the VLDB Endowment*, 15(10): 2216–2229.

Tang, Y.; Shi, Y.; and Xiao, X. 2015. Influence maximization in near-linear time: A martingale approach. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, 1539–1554.

Tang, Y.; Xiao, X.; and Shi, Y. 2014. Influence maximization: Near-optimal time complexity meets practical efficiency. In *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, 75–86.

Vaswani, S.; Kveton, B.; Wen, Z.; Ghavamzadeh, M.; Lakshmanan, L. V.; and Schmidt, M. 2017. Model-independent online learning for influence maximization. In *International Conference on Machine Learning*, 3530–3539.

Wang, Y.; Fan, Q.; Li, Y.; and Tan, K.-L. 2017. Real-time influence maximization on dynamic social streams. *Proceedings of the VLDB Endowment*, 10(7): 805–816.

Wilder, B.; Onasch-Vera, L.; Hudson, J.; Luna, J.; Wilson, N.; Petering, R.; Woo, D.; Tambe, M.; and Rice, E. 2018. End-to-End influence maximization in the field. In *Proceedings of the 17th International Conference on Autonomous Agents and Multi-Agent Systems*, 1414–1422.

Yu, W.; DeLellis, P.; Chen, G.; di Bernardo, M.; and Kurths, J. 2012. Distributed adaptive control of synchronization in complex networks. *IEEE Transactions on Automatic Control*, 57(8): 2153–2158.

Zhang, J.; Liu, B.; Tang, J.; Chen, T.; and Li, J. 2013. Social influence locality for modeling retweeting behaviors. In *International Joint Conference on Artificial Intelligence*, volume 13, 2761–2767.