Scalable Misinformation Prevention in Social Networks

Michael Simpson, Venkatesh Srinivasan, and Alex Thomo
Computer Science Dept., University of Victoria, B.C., Canada
{simpsonm,srinivas,thomo}@uvic.ca

ABSTRACT

In this work, we consider misinformation propagating through a social network and study the problem of its prevention. In this problem, a "bad" campaign starts propagating from a set of seed nodes in the network and we use the notion of a limiting (or "good") campaign to counteract the effect of misinformation. The goal is then to identify a subset of k users that need to be convinced to adopt the limiting campaign so as to minimize the number of people that adopt the "bad" campaign at the end of both propagation processes.

This work presents *RPS* (Reverse Prevention Sampling), an algorithm that provides a scalable solution to the misinformation prevention problem. Our theoretical analysis shows that *RPS* runs in $O((k+l)(n+m)(\frac{1}{1-\gamma})\log n/\epsilon^2)$ expected time and returns a $(1-1/e-\epsilon)$-approximate solution with at least $1-n^{-l}$ probability (where γ is a typically small network parameter). The time complexity of *RPS* substantially improves upon the previously best-known algorithms that run in time $\Omega(mnk \cdot POLY(\epsilon^{-1}))$. We experimentally evaluate *RPS* on large datasets and show that it outperforms the state-of-the-art solution by several orders of magnitude in terms of running time. This demonstrates that misinformation prevention can be made practical while still offering strong theoretical guarantees.

PVLDB Reference Format:
Michael Simpson, Venkatesh Srinivasan, Alex Thomo. Scalable Misinformation Prevention in Social Networks. *PVLDB*, 12(xxx): xxxx-yyyy, 2019.
DOI: https://doi.org/TBD

1. INTRODUCTION

Social networks allow for the widespread distribution of knowledge and information in modern society as they are rapidly becoming a place for people to hear the news and discuss social topics. Information can spread quickly through the network eventually reaching a large audience, especially so for influential users. However, while the ease of information propagation in social networks can be beneficial, it can also have disruptive effects. In recent years, the number of high profile instances of misinformation causing severe real-world effects has risen sharply. These examples range across a number of social media platforms and topics. A series of bogus tweets from a trusted news network referring to explosions at the White House caused immediate and extensive repercussions in the financial markets [8]. During a recent shooting at YouTube's headquarters, and before police managed to secure the area, a wave of misinformation and erroneous accusations were widely disseminated on Twitter causing panic and confusion [21, 10]. Finally, there has been much discussion on the role misinformation and fake news played in the 2016 U.S. presidential election with sites such as Reddit and Facebook being accused of harbouring and spreading divisive content and misinformation [11, 23, 1]. Thus, in order for social networks to serve as a reliable platform for disseminating critical information, it is necessary to have tools to limit the spread of misinformation.

Budak et al. [4] were the first to formulate the problem of misinformation prevention as a combinatorial optimization problem. By building upon the seminal work of Kempe et al. [15] on *influence maximization* for the single campaign model to handle multiple campaigns ("bad" and "good"), they present a greedy approach that provides a $(1-1/e-\epsilon)$-approximate solution. Unfortunately the greedy approach of [4] is plagued by the same scaling issues as [15] when considering large social networks and is further exacerbated by the added complexity of tracking multiple cascades which requires costly shortest paths computations. This leads us to the motivating question for this paper: Can we find scalable algorithms for the misinformation prevention problem introduced in [4]?

The scalability hurdle for the case of a single campaign was recently resolved by Borgs et al. [3] when they made a theoretical breakthrough that fundamentally shifts the way in which we view the influence maximization problem. Their key insight was to reverse the question of "what subset of the network can a particular user influence" to "who could have influenced a particular user". Their sampling method runs in close to linear time and returns a $(1-1/e-\epsilon)$-approximate solution with at least $1-n^{-l}$ probability. In addition, Tang et al. [24] presented a significant advance that improved the practical efficiency of the work by Borgs et al. through a careful theoretical analysis that rids the approach of [3] of a large hidden constant in the runtime guarantee.

Borgs et al. [3] leave open the question whether their techniques can be extended to other influence propagation models. In this work, we resolve the question of [3] for the mis-

Proceedings of the VLDB Endowment, Vol. 12, No. xxx
Copyright 2018 VLDB Endowment 2150-8097/18/10... $ 10.00.
DOI: https://doi.org/TBD

information prevention problem and achieve scalability in the multi-campaign model. We complement our theoretical analysis with extensive experiments which show an improvement of several orders of magnitude over Budak et al. [4].

Since influence in the single campaign setting corresponds to reachability in the network, our solution requires mapping the concept of reachability to an analogous notion in the multi-campaign model of [4]. In [4] they introduce shortest paths as the metric to determine the set of nodes that can be saved from adopting the misinformation (i.e. influenced by the "good" information). Our first contribution is to identify that shortest paths alone are not sufficient in determining the ability to save a particular node in the network and introduce the crucial notion of *blocked* nodes.

Using our newly defined notion of blocking, we generalize the sampling-based approach of Borgs et al. to the multi-campaign setting. A major contribution of this work, and critical difference from the approach of [3], is a modified breadth-first search we design to compute the set of nodes that could have saved a particular user from adopting the misinformation. Our novel algorithm handles the notion of blocked nodes when traversing in the transpose graph by utilizing detailed bookkeeping techniques to effectively track those nodes that would become blocked.

Furthermore, we are able to adapt optimization ideas from Tang et al. to further improve the scalability of our approach. In particular, we show that our algorithm can efficiently handle graphs with more than 50 million edges. As a consequence, we are able to match the progress of both [3] and [24] together in this work.

Finally, we obtain theoretical guarantees on the expected runtime and solution quality for our new approach and show that its expected runtime substantially improves upon the expected runtime of [4]. Additionally, we rule out sublinear algorithms for our problem through a lower bound on the time required to obtain a constant approximation.

In summary, the contributions of this paper are:

1. We introduce the concept of blocked nodes that fully captures the requirements necessary for preventing the adoption of misinformation in the multi-campaign model.
2. We design and implement a novel BFS-based algorithm for computing the set of nodes that could save a particular user from adopting the misinformation.
3. We propose a misinformation prevention approach which returns a $(1 - 1/e - \epsilon)$-approximate solution with high probability in the multi-campaign model and show that its expected runtime substantially improves upon the expected runtime of the algorithm of Budak et al. [4].
4. We give a lower bound of $\Omega(m+n)$ on the time required to obtain a constant approximation for the misinformation prevention problem.
5. We experiment with large datasets and show that our algorithm gives an improvement of several orders of magnitude over Budak et al. [4] and can efficiently handle graphs with more than 50 million edges.

2. PRELIMINARIES

In this section, we formally define the multi-campaign diffusion model, the eventual influence limitation problem presented by Budak et al. [4], and present an overview of Kempe et al. and Borgs et al.'s work [15, 3] on the influence maximization problem.

2.1 Diffusion Model

Let $\mathcal{G}$ be a social network with a node set V and a directed edge set E, with $|V| = n$ and $|E| = m$. Let C (for "bad *C*ampaign") and L (for "*L*imiting") denote two influence campaigns. Assume that each directed edge e in $\mathcal{G}$ is associated with propagation probabilities $p_C(e), p_L(e) \in [0,1]$. Further, let G be the underlying unweighted graph (ignoring edge probabilities). Given $\mathcal{G}$, the Multi-Campaign Independent Cascade model (MCIC) of Budak et al. [4] considers a time-stamped influence propagation process as follows:

1. At timestamp 1, we *activate* selected sets A_C and A_L of nodes in $\mathcal{G}$ for campaigns C and L respectively, while setting all other nodes *inactive*.
2. If a node u is first activated at timestamp i in campaign C (or L), then for each directed edge e that points from u to an inactive neighbour v in C (or L), u has $p_C(e)$ (or $p_L(e)$) probability to activate v at timestamp $i+1$. After timestamp $i+1$, u cannot activate any node.
3. In the case when two or more nodes from different campaigns are trying to activate v at a given time step we assume that the "good information" (i.e. campaign L) takes effect.
4. Once a node becomes activated in one campaign, it never becomes inactive or changes campaigns.

2.2 Eventual Influence Limitation

A natural objective function, as outlined in [4], is "saving" as many nodes as possible. That is, we seek to minimize the number of nodes that end up adopting campaign C when the propagation process is complete. This is referred to as the *eventual influence limitation problem (EIL)*.

Let A_C and A_L be the set of nodes from which campaigns C and L start, respectively. Let $I(A_C)$ be the set of nodes that are activated in campaign C in the absence of L when the above propagation process converges and $\pi(A_L)$ be the size of the subset of $I(A_C)$ that campaign L prevents from adopting campaign C. We refer to A_L and A_C as the *seed sets*, $I(A_C)$ as the *influence* of campaign C, and $\pi(A_L)$ as the *prevention* of campaign L. The nodes that are prevented from adopting campaign C are referred to as *saved*. Note that $\pi(A_L)$ is a random variable that depends on the edge probabilities that each node uses in determining out-neighbors to activate.

Now, notice that the above process has the following equivalent description. We can interpret $\mathcal{G}$ as a distribution over unweighted directed graphs, where each edge e is independently realized with probability $p_C(e)$ (or $p_L(e)$). If we realize a graph g according to this probability distribution, then we can associate the set of saved nodes in the original process with the set of nodes which campaign L reaches before campaign C during the diffusion process in $g \sim \mathcal{G}$. We will make use of this alternative formulation of the MCIC model throughout the paper.

Budak et al. [4] present a simplified and natural version of the problem where the information from campaign L is accepted by users with probability 1 ($p_L(e) = 1$ if edge e exists and $p_L(e) = 0$ otherwise). This is referred to as the *high effectiveness property*. In [4] it is shown that EIL with the high effectiveness property is NP-hard. Interestingly, with the high effectiveness property, the prevention function is submodular and thus the greedy approach yields approximation guarantees. Budak et al. study and obtain results for this version of the EIL problem and is the problem that we consider here.

Further, as in [4], we assume that the spread of influence for campaign C starts from a single node v_c, referred to as the *adversary node*. However, note that our approach can be easily extended to the case where C starts from a set of nodes. Additionally, our approach works when we allow campaign C to be detected with delay Δ at which time campaign L is initiated.

Problem Statement. Given $\mathcal{G}$, seed set A_C, and a positive integer k, the eventual influence limitation (EIL) problem asks for a size-k seed set A_L maximizing the value of $\mathbb{E}[\pi(A_L)]$.

2.3 Influence Maximization Problem

In the following we outline two approaches to the well studied *influence maximization problem (IM)*. This problem is posed in the popular Independent Cascade model (IC) which, unlike the MCIC model, only considers a single campaign. The goal here is to compute a seed set S of size k that maximizes the influence of S in $\mathcal{G}$.

2.3.1 Kempe et al.'s Greedy Approach

Kempe et al.'s approach [15] to the IM problem (referred to as *Greedy* in the following) builds up the seed set S in k iterations by adding one node to S in each round. In each iteration it adds into S the node u that gives the largest marginal increase in $\mathbb{E}[I(S)]$.

However, the computation of $\mathbb{E}[I(S)]$ is #P-hard. Therefore, the node u leading to the largest marginal increase in influence must be estimated via a sufficient number of Monte Carlo simulations. In [14] it is shown that if we take a large number $r \in \Omega(POLY(\frac{n}{\epsilon}))$ of measurements in the estimation of each $\mathbb{E}[I(S)]$, then, with high probability, *Greedy* yields a $(1-1/e-\epsilon)$-approximate solution in the IC model.

Note, if we let $R_H(S)$ be the set of nodes in a graph H that are *reachable* from S (a node v in H is reachable from S if there exists a directed path in H that starts from a node in S and ends at v), then $I(S) = R_g(S)$ for a fixed $g \sim \mathcal{G}$.

Despite *Greedy*'s simplicity, its ability to scale to large social networks is severely restricted by its $\Omega(mnk \cdot POLY(\epsilon^{-1}))$ time complexity. As a result, *Greedy* runs for days even when the network contains only a few thousand nodes [5].

2.3.2 Borgs et al.'s Method

Borgs et al. [3] propose a novel method for solving the IM problem under the IC model that avoids the limitations of *Greedy*. We follow the convention of [24] and refer to the method of [3] as *Reverse Influence Sampling (RIS)*. To explain how *RIS* works, Tang et al. [24] introduce the following definitions:

Definition 1 (Reverse Reachable Set) *The reverse reachable (RR) set for a node v in $g \sim \mathcal{G}$ is the set of nodes that can reach v. (That is, for each node u in the RR set, there is a directed path from u to v in g.)*

Definition 2 (Random RR Set) *A random RR set is an RR set generated on an instance of $g \sim \mathcal{G}$, for a node selected uniformly at random from g.*

The connection between RR sets and node activations is formalized in the following lemma.

LEMMA 1. [3] *For any seed set S and node v, the probability that an influence propagation process from S can activate v equals the probability that S overlaps an RR set for v.*

Note that an RR set for a node v is generated by sampling a $g \sim \mathcal{G}$. Based on this result, Borgs' et al. *RIS* algorithm runs in two steps

1. Generate random RR sets from $\mathcal{G}$ until a threshold on the total number of steps taken has been reached.

2. Consider the maximum coverage problem of selecting k nodes to cover the maximum number of RR sets generated. Use the standard greedy algorithm for the problem to derive a $(1-1/e)$-approximate solution S_k^*. Return S_k^* as the seed set to use for activation.

The rationale behind *RIS* is as follows: if a node u appears in a large number of RR sets it should have a high probability to activate many nodes under the IC model; hence, u's expected influence should be large. As such, we can think of the number of RR sets u appears in as an estimator for u's expected influence. By the same reasoning, if a size-k node set S_k^* covers most RR sets, then S_k^* is likely to have the maximum expected influence among all size-k node sets in $\mathcal{G}$ leading to a good solution to the IM problem. Therefore, the primary goal is to show that we have a *good* estimator for u's expected influence. The main contribution of Borgs et al. is an analysis of their proposed threshold-based approach: they allow *RIS* to keep generating RR sets, until the total number of nodes and edges examined during the generation process reaches a pre-defined threshold Γ. They show that when Γ is set to $\Theta((m+n)k \log n/\epsilon^2)$, *RIS* runs in time $O((m+n)k \log n/\epsilon^2)$, and it returns a $(1-1/e-\epsilon)$-approximate solution to the IM problem with at least constant probability.

3. NEW DEFINITIONS

In this section we introduce new definitions that are crucial to our approach. Importantly, we present and discuss a gap in the work of Budak et al. that requires closing in order for our approach to be possible.

Given set A_L of vertices and (unweighted) directed graph $g \sim \mathcal{G}$, write $cl_g(A_L)$ for the set of nodes closer to A_L than $A_C = \{v_c\}$ in g. That is, a node $w \in cl_g(A_L)$ if there exists a node v such that $v \in A_L$ and $|SP_G(v,w)| \leq |SP_g(v_c,w)|$ where $SP_H(v,w)$ denotes a shortest path from node v to w in graph H. When g is drawn from $\mathcal{G}$ this is a necessary, but not sufficient[1], condition for the set of nodes *saved* by A_L. We also require that the nodes in $cl_g(A_L)$ not be *blocked* by the diffusion of campaign C in g.

Definition 3 (Blocked Nodes) *A node $w \in cl_g(A_L)$ is* blocked *and cannot be saved by A_L if for every path p from A_L to w there exists a node u such that $|SP_g(v_c,u)| < |SP_G(A_L,u)|$.*

[1]In Budak et al.'s work, the set of nodes closer to A_L than A_C is established as a necessary and sufficient condition to *save* a node in the MCIC model, but we note that this should be revised to include the *blocking* condition due to a gap in the proof of Claim 1 in [4].

Notation	Description
$\mathcal{G}$	a social network represented as a weighted directed graph $\mathcal{G}$
G, G_T	the underlying unweighted graph G and its transpose G_T constructed by reversing the direction of each edge
n, m	the number of nodes and edges in $\mathcal{G}$ respectively
k	the size of the seed set for misinformation prevention
C, L	the misinformation campaign C and the limiting campaign L
$p_C(e)$, $p_L(e)$	the propagation probability on an edge e for campaigns C and L respectively
$\pi(S)$	the prevention of a node set S in a misinformation propagation process on $\mathcal{G}$ (see Section 3)
$\omega(R)$, $\omega_\pi(R)$	the number of edges considered in generating an RRC set and that originate from nodes in an RRC set R (see Equation 1)
$\kappa(R)$	see Equation 9
$\mathcal{R}$	the set of all RRC sets generated by Algorithm 1
$\mathcal{F}_\mathcal{R}(S)$	the fraction of RRC sets in $\mathcal{R}$ that are covered by a node set S
EPT	the expected width of a random RRC set
EXP_C	the expected influence of campaign C
OPT_L	the maximum $\pi(S)$ for any size-k seed set S
KPT	a lower bouund of OPT_L established in Section 4.2
λ	see Equation 4

Table 1: Frequently used notation.

Let $bl_g(A_L)$ be the set of blocked nodes for A_L. Conceptually, the nodes in $bl_g(A_L)$ are cutoff because some node on the paths from A_L is reached by campaign C before L which stops the diffusion of L.

To help illustrate the concept of blocked nodes, consider the graph presented in Figure 1. Assume that the solid lines are *live* edges that make up graph $g \sim \mathcal{G}$ in the influence propagation process. Then, the dotted lines are edges that were not realized for campaign C. The adversary campaign C starts from v_c while the limiting campaign L starts from v.

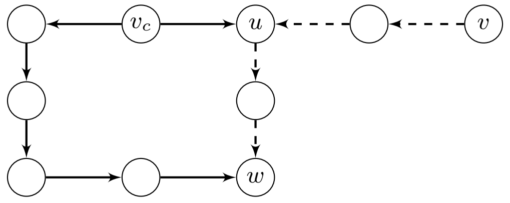


Figure 1: An example illustrating the concept of blocked nodes.

Observe that $|SP_G(v, w)| = 4$ and $|SP_g(v_c, w)| = 5$. However, w cannot be saved in the resulting cascade since at timestamp 1 the node u will adopt campaign C. This intersects the shortest path from v to w and therefore campaign L will not be able to reach node w since a node never switches campaigns. Thus, we say that node w is *blocked* by C.

Importantly, the approach of [4] can be recovered with a modified proof for Claim 1 and Theorem 4.2 of [4]. The statements must include the notion of blocked nodes in their *inoculation graph* definition, but a careful inspection shows that their objective function remains submodular after this inclusion. As a result, the greedy approach still provides the stated approximation guarantees and also allows us to incorporate the ideas of [3] in our solution (as they require a submodular objective function).

Next, we formally define the prevention, $\pi(A_L)$, which corresponds to the number of nodes saved by A_L. That is, $\pi(A_L) = |R_g(v_c) \cap (cl_g(A_L) \setminus bl_g(A_L))|$. We write $\mathbb{E}[\pi(A_L)] = \mathbb{E}_{g\sim\mathcal{G}}[\pi(A_L)]$ for the expected prevention of A_L in $\mathcal{G}$.

We refer to the set of nodes that could have saved u as the *saviours* of u. A node w is a candidate saviour for u if there is a directed path from w to u in G (i.e. reverse reachability). Then, w is a saviour for u subject to the additional constraint that w would not be cutoff by the diffusion of A_C in g. That is, a candidate saviour w would be cutoff and cannot be a saviour for u if for every path p from w to u there exists a node v_b such that $|SP_g(v_c, v_b)| < |SP_G(w, v_b)|$. We refer to the set of candidate saviours for u that are cutoff as $\tau_g(u)$. Thus, we can define the saviours of u as the set $R_{G^T}(u) \setminus \tau_g(u)$. Therefore, we have:

Definition 4 (Reverse Reachability without Cutoff Set) *The reverse reachability without cutoff (RRC) set for a node v in $g \sim \mathcal{G}$ is the set of saviour nodes of v, i.e. the set of nodes that can save v. (That is, for each node u in the RRC set, $u \in R_{G^T}(u) \setminus \tau_g(u)$.) Note, if $v \notin I(v_c)$ then we define the corresponding RRC set as empty since v is not eligible to be saved.*

Definition 5 (Random RRC Set) *A random RRC set is an RRC set generated on an instance of $g \sim \mathcal{G}$, for a node selected uniformly at random from g.*

Finally, let $OPT_L = max_{S:|S|=k}\{\mathbb{E}[\pi(S)]\}$ be the maximum expected prevention of a set of k nodes.

4. REVERSE PREVENTION SAMPLING

This section presents our misinformation prevention method, *Reverse Prevention Sampling (RPS)*, that applies the *RIS* approach to the EIL problem presented in the MCIC model of Budak et al. [4]. Our approach borrows ideas from the work of Tang et al. [24] which improves the practical efficiency of Borgs et al. approach. Our algorithm returns a $(1 - \frac{1}{e} - \epsilon)$-approximation to the EIL problem, with success probability $1 - n^{-l}$, in time $O((k+l)(m+n)(\frac{1}{1-\gamma}) \log n/\epsilon^2)$. At a high level, *RPS* consists of two steps.

1. **Parameter Estimation.** Compute a lower-bound for the maximum expected prevention among all possible size-k seed sets for A_L and then use the lower-bound to derive a parameter θ.

2. **Node Selection.** Sample θ random RRC sets from $\mathcal{G}$ to form a set $\mathcal{R}$ and then compute a size-k seed set S_k^* that covers a large number of RRC sets in $\mathcal{R}$. Return S_k^* as the final result.

We briefly remark on the difficulty of deriving θ as it must be larger than a particular threshold necessary for ensuring the correctness of *RPS*, but the threshold depends on the optimal prevention, which is an unknown quantity. In our parameter estimation step we derive a θ value for *RPS* that is above the threshold but also allows for practical efficiency.

In the rest of this section we first identify the conditions necessary for the node selection phase of *RPS* to return a good solution and then describe the the parameter estimation phase in detail. Table 1 provides a quick reference to the frequently used notation. Finally, due to space constraints, we omit some proofs and only include those that help build the intuition behind our approach or aid the exposition.

4.1 Node Selection

The pseudocode of *RPS*'s node selection step is presented in Algorithm 1. Given $\mathcal{G}$, k, A_C, and a constant θ as input, the algorithm stochastically generates θ random RRC sets, accomplished by repeated invocation of the prevention of misinformation process, and inserts them into a set $\mathcal{R}$. Next, the algorithm follows a greedy approach for the *maximum coverage problem* to select the final seed set. In each iteration, the algorithm selects a node v_i that covers the largest number of RRC sets in $\mathcal{R}$, and then removes all those covered RRC sets from $\mathcal{R}$. The k selected nodes are put into a set S_k^*, which is returned as the final result.

Algorithm 1 NodeSelection($\mathcal{G}$,k,A_C,θ)

```
1: $\mathcal{R} \leftarrow \emptyset$
2: Generate $\theta$ random RRC sets and insert them into $\mathcal{R}$.
3: Initialize a node set $S_k^* \leftarrow \emptyset$
4: for $i = 1,\ldots,k$ do
5:     Identify the node $v_i$ that covers the most RRC sets in $\mathcal{R}$
6:     Add $v_i$ into $S_k^*$
7:     Remove from $\mathcal{R}$ all RRC sets that are covered by $v_i$
8: return $S_k^*$
```

Lines 4-8 in Algorithm 1 correspond to a standard greedy approach for a *maximum coverage problem.* The problem is equivalent to maximizing a submodular function with cardinality constraints for which it is well known that a greedy approach returns a $(1 - 1/e)$-approximate solution in linear time [19].

RRC set generation. Line 2 generates $\mathcal{R}$ by repeated simulation of the misinformation prevention process. The generation of each random RRC set is implemented as two breath-first searches (BFS) on $\mathcal{G}$ and G^T respectively. The first BFS on $\mathcal{G}$ corresponds to a standard randomized BFS and computes the influence set of A_C. The second BFS on G^T is an important algorithmic contribution of this work that is essential for the *RPS* approach. This novel bounded-depth BFS with pruning carefully tracks which nodes will become blocked and is described in detail below.

The set of nodes traversed by the standard randomized BFS on $\mathcal{G}$ is equivalent to $I(v_c)$ for $g \sim \mathcal{G}$, due to deferred randomness. Note that in each step of the BFS we record at each node w the distance from v_c to w, denoted $D(w)$, for use in the second step.

Given a randomly selected node u in G, observe that in order for u to be able to be saved we require $u \in I(v_c)$. Therefore, if the randomly selected node $u \notin I(v_c)$ then we return an empty RRC set. On the other hand, if $u \in I(v_c)$, we have $D(u) = |SP_g(v_c, u)|$ as a result of the above randomized BFS which indicates the maximum distance from u that candidate saviour nodes can exist. We run a second BFS from u in G^T to depth $D(u)$ to determine the saviour nodes for u by carefully pruning those nodes that would become blocked.

The *bounded-depth BFS with pruning* on G^T, presented in Algorithm 2, takes as input a source node u, the maximum depth $D(u)$, and a directed graph G^T. Algorithm 2 utilizes special indicator values associated with each node w to account for potential cutoffs from v_c. Each node w holds a variable, $\beta(w)$, which indicates the distance beyond w that the BFS can go before the diffusion would have been cutoff by v_c in g. The β value for each node w is initialized to $D(w)$. In each round, the current node w has an opportunity to update the β value of each of its successors only if $\beta(w) > 0$. For each successor z of w, we assign $\beta(z) = \beta(w) - 1$ if $\beta(z) =$ `null` or if $\beta(z) > 0$ and $\beta(w) - 1 < \beta(z)$. In this way, each ancestor of z will have an opportunity to apply a β value to z to ensure that if any ancestor has a β value then so will z and furthermore, the β variable for z will be updated with the smallest β value from its ancestors. We terminate the BFS early if we reach a node w with $\beta(w) = 0$.

Figure 2 captures the primary scenarios encountered by Algorithm 2 when initialized at node u. The enclosing dotted line represents the extent of the influence of campaign C for the current influence propagation process. First, notice that if the BFS moves away from v_c, as in the case of node z, that, once we move beyond the influence boundary of v_c, there will be no potential for cutoff. As such, the BFS is free to traverse until the maximum depth $D(u)$ is reached. On the other hand, if the BFS moves towards (or perpendicular to) v_c then we must carefully account for potential cutoff. For example, when the BFS reaches v, we know the distance from v_c to v: $D(v) = SP_g(v_c, v)$. Therefore, the BFS must track the fact that there cannot exist saviours at a distance $D(v)$ beyond v. In other words, if we imagine initializing a misinformation prevention process from a node w such that $SP_G(v, w) > D(v)$ then v will adopt campaign C before campaign L can reach v. Therefore, at each out-neighbour of v we use the knowledge of $D(v)$ to track the distance beyond v that saviours can exist. This updating process tracks the smallest such value and is allowed to cross the enclosing influence boundary of campaign C ensuring that all potential for cutoff is tracked.

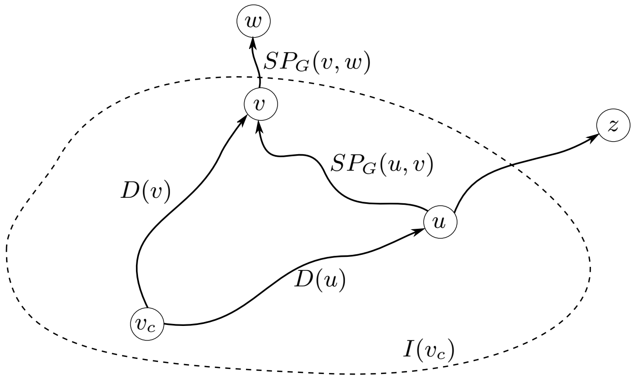


Figure 2: An example illustrating the concept of blocked nodes.

Finally, we collect all nodes visited during the process (including u), and use them to form an RRC set. The runtime of this procedure is precisely the sum of the degrees (in G) of the nodes in $I(v_c)$ plus the sum of the degrees of the nodes in $R_{G^T}(u) \setminus \tau(u)$.

Algorithm 2 generateRRC(u, $D(u)$, G^T)

```
1: let $Q$ be a queue
2: $u.depth = 0$
3: $Q.enqueue(u)$
4: label $u$ as discovered
5: while $Q$ is not empty do
6:     $w \leftarrow Q.dequeue()$
7:     if $w.depth = D(u)$ OR $\beta(w) = 0$ then
8:         continue
9:     for all nodes $z$ in $G^T.adjacentEdges(w)$ do
10:        if $\beta(w) > 0$ AND $\beta(z) > 0$ then
11:            if $\beta(w) - 1 < \beta(z)$ then
12:                $\beta(z) \leftarrow \beta(w) - 1$
13:        else if $\beta(w) > 0$ then
14:            $\beta(z) \leftarrow \beta(w) - 1$
15:        if $z$ is not labelled as discovered then
16:            $z.depth = w.depth + 1$
17:            $Q.enqueue(z)$
18:            label $z$ as discovered
```

Performance Bounds. In this section we first define a quantity EPT that captures the expected number of edges traversed when generating a random RRC set. After that, we define the expected runtime of *RPS* in terms of EPT and the parameter θ. Finally, we define a threshold for θ that ensures the correctness of *RPS*.

First, we refer to the instance of $I(v_c)$ used in computing an RRC set R as M_R. Then, we define the *width* of an RRC set R, denoted as $\omega(R)$, as the number of edges in G that point to nodes in R plus the number of edges in G that originate from nodes in M_R. That is

$$\omega(R) = \sum_{u \in M_R} outdegree_G(u) + \sum_{v \in R} indegree_G(v) \qquad (1)$$

Furthermore, we define the *subwidth* of an RRC set R, denoted as $\omega_\pi(R)$, as the number of edges in G that point to nodes in R. That is $\omega_\pi(R) = \sum_{v \in R} indegree_G(v)$. Also, we define the *prewidth* of an RRC set R, denoted as $\omega(M_R)$, as the number of edges in G that originate from nodes in M_R. That is $\omega(M_R) = \sum_{u \in M_R} outdegree_G(u)$.

Let EPT be the expected width of a random RRC set, where the expectation is taken over the randomness in R and M_R, and observe that Algorithm 1 has an expected runtime of $O(\theta \cdot EPT)$. This can be observed by noting that EPT captures the expected number of edge traversals required to generate a random RRC set since an edge is only considered in the propagation process (either of the two BFS's) if it points to a node in R or originates from a node in M_R. The remainder of the section aims to derive a value for θ that minimizes the expected runtime of Algorithm 1 while ensuring the quality of the solution returned.

We establish a connection between RRC sets and the prevention process on $\mathcal{G}$. That is, the prevention of a set of nodes S is precisely n times the probability that a node u, chosen uniformly at random, has a saviour from S:

Lemma 2. *For any seed set S and any node v, the probability that a prevention process from S can save v equals the probability that S overlaps an RRC set for v.*

Proof. Let S be a fixed set of nodes, and v be a fixed node. Suppose that we generate an RRC set R for v on a graph $g \sim \mathcal{G}$. Let ρ_1 be the probability that S overlaps with R and let ρ_2 be the probability that S, when used as a seed set, can save v in a prevention process on $\mathcal{G}$. By Definition 4, if $v \in I(v_c)$ then ρ_1 equals the probability that a node $u \in S$ is a saviour for v. That is, ρ_1 equals the probability that G contains a directed path from $u \in S$ to v and $u \notin \tau(v)$ and 0 if $v \notin I(v_c)$. Meanwhile, if $v \in I(v_c)$ then ρ_2 equals the probability that a node $u \in S$ can save v (i.e. $v \in (cl_g(u) \setminus bl_g(u))$) and 0 if $v \notin I(v_c)$. It follows that $\rho_1 = \rho_2$ due to the symmetry between the set of saviours for v and the ability to save v. $\square$

For any node set S, let $F_\mathcal{R}(S)$ be the fraction of RRC sets in $\mathcal{R}$ covered by S. Then, based on Lemma 2, we can prove that the expected value of $n \cdot F_\mathcal{R}(S)$ equals the expected prevention of S in $\mathcal{G}$.

Corollary 1. $\mathbb{E}[n \cdot F_\mathcal{R}(S)] = \mathbb{E}[\pi(S)]$

Corollary 1 implies that we can estimate $\mathbb{E}[\pi(S)]$ by estimating the fraction of RRC sets in $\mathcal{R}$ covered by S. The number of sets covered by a node v in $\mathcal{R}$ is precisely the number of times we observed that v was a saviour for a randomly selected node u. We can therefore think of $n \cdot F_\mathcal{R}(S)$ as an estimator for $\mathbb{E}[\pi(S)]$. Our primary task is to show that it is a *good* estimator. Using Chernoff bounds, we show that $n \cdot F_\mathcal{R}(S)$ is an accurate estimator of any node set S's expected prevention, when θ is sufficiently large:

Lemma 3. *Suppose that θ satisfies*

$$\theta \geq (8 + 2\epsilon)n \cdot \frac{l \log n + \log \binom{n}{k} + \log 2}{OPT_L \cdot \epsilon^2} \qquad (2)$$

Then, for any set S of at most k nodes, the following inequality holds with at least $1 - n^{-l}/\binom{n}{k}$ probability:

$$\left| n \cdot F_\mathcal{R}(S) - \mathbb{E}[\pi(S)] \right| < \frac{\epsilon}{2} \cdot OPT_L \qquad (3)$$

Based on Lemma 3, we prove that if Eqn. 2 holds, Algorithm 1 returns a $(1 - 1/e - \epsilon)$-approximate solution with high probability by a simple application of Chernoff bounds.

Theorem 1. *Given a θ that satisfies Equation 2, Algorithm 1 returns a $(1 - 1/e - \epsilon)$-approximate solution with at least $1 - n^{-l}/\binom{n}{k}$ probability.*

An important consideration is that, since OPT is unknown, we can not set θ directly from Equation 2. For simplicity, we define

$$\lambda = (8 + 2\epsilon)n \cdot \left(l \log n + \log \binom{n}{k} + \log 2\right) \cdot \epsilon^{-2} \qquad (4)$$

and rewrite Equation 2 as

$$\theta \geq \lambda / OPT_L \qquad (5)$$

The following section details how to derive a θ which not only satisfies Equation 2, but also leads to an $O((k+l)(m+n)(\frac{1}{1-\gamma}) \log n / \epsilon^2)$ expected runtime for Algorithm 1.

4.2 Parameter Estimation

Our objective is to identify a θ that makes $\theta \cdot EPT$ reasonably small, while still ensuring $\theta \geq \lambda/OPT_L$. First, we define a probability distribution $\mathcal{V}^*$ over the nodes in G, such that the probability mass for each node is proportional to its indegree in G. Let v^* be a random variable following $\mathcal{V}^*$ and recall that M_R is a random instance of $I(v_c)$.

LEMMA 4. *$\frac{m}{n} \cdot \mathbb{E}[\pi(\{v^*\})] = EPT - \mathbb{E}[\omega(M_R)]$, where the expectation of $\pi(\{v^*\})$ and $\omega(M_R)$ is taken over the randomness in v^* and the prevention process.*

PROOF. Let R be a random RRC set, M_R be the random instance of $I(v_c)$ used to compute R, and p_R be the probability that a randomly selected edge from G points to a node in R. Then, $EPT = \mathbb{E}[\omega(M_R)] + \mathbb{E}[p_R \cdot m]$, where the expectation is taken over the random choices of R and M_R.

Let v^* be a sample from $\mathcal{V}^*$ and $b(v^*, R)$ be a boolean function that returns 1 if $v^* \in R$, and 0 otherwise. Then, for any fixed R, $p_R = \sum_{v^*}(\Pr[v^*] \cdot b(v^*, R))$. Now consider that we fix v^* and vary R. Define $p_{v^*,R} = \sum_R(\Pr[R] \cdot b(v^*, R))$ so that by Lemma 2, $p_{v^*,R}$ equals the probability that a randomly selected node can be saved in a prevention process when $\{v^*\}$ is used as a seed set. Therefore, $\mathbb{E}[p_{v^*,R}] = \mathbb{E}[\pi(\{v^*\})]/n$. This leads to

$$\begin{aligned}\mathbb{E}[p_R] &= \sum_R(\Pr[R] \cdot p_R)\\ &= \sum_R(\Pr[R] \cdot \sum_{v^*}(\Pr[v^*] \cdot b(v^*, R)))\\ &= \sum_{v^*}(\Pr[v^*] \cdot \sum_R(\Pr[R] \cdot b(v^*, R)))\\ &= \sum_{v^*}(\Pr[v^*] \cdot p_{v^*,R}) = \mathbb{E}[p_{v^*,R}] = \mathbb{E}[\pi(\{v^*\})]/n\end{aligned}$$

Resulting in

$$\begin{aligned}EPT &= \mathbb{E}[\omega(M_R)] + m \cdot \mathbb{E}[p_R]\\ &= \mathbb{E}[\omega(M_R)] + \frac{m}{n} \cdot \mathbb{E}[\pi(\{v^*\})]\end{aligned}$$

This completes the proof. $\square$

Lemma 4 shows that if we randomly sample a node from $\mathcal{V}^*$ and calculate its expected prevention p, then on average we have $p = \frac{n}{m}(EPT - \mathbb{E}[\omega(M_R)])$. This implies that $\frac{n}{m}(EPT - \mathbb{E}[\omega(M_R)]) \leq OPT_L$, since OPT_L is the maximum expected prevention of any size-k node set.

Recall that the expected runtime complexity of Algorithm 1 is $O(\theta \cdot EPT)$. Now, suppose we are able to identify a parameter t such that $t = \Omega(\frac{n}{m}(EPT - \mathbb{E}[\omega(M_R)]))$ and $t \leq OPT_L$. Then, by setting $\theta = \lambda/t$, we can guarantee that Algorithm 1 is correct, since $\theta \geq \lambda/OPT_L$, and has an expected runtime complexity of

$$O(\theta \cdot EPT) = O\Big(\frac{\lambda}{t} \cdot EPT\Big) = O\left(\frac{\lambda \cdot EPT}{\frac{n}{m}(EPT - \mathbb{E}[\omega(M_R)])}\right) \tag{6}$$

Furthermore, we can define a ratio $\gamma \in (0, 1)$ which captures the relationship between $\mathbb{E}[\omega(M_R)]$ and EPT by writing $\mathbb{E}[\omega(M_R)] = \gamma EPT$. Then we can rewrite Equation 6 as

$$O\Big(\frac{m}{n}\Big(\frac{1}{1-\gamma}\Big)\lambda\Big) = O((k + l)(m + n)(1/(1 - \gamma)) \log n/\epsilon^2) \tag{7}$$

Note that γ is an instance-based parameter not present in the work of Tang et al, but arises from the MCIC model. See Section 5 for a detailed discussion of γ.

Computing t. Ideally, we seek a t that increases monotonically with k to mimic the behaviour of OPT_L. Suppose we take k samples $e_i = (v_i, w_i)$ with replacement over a uniform distribution on the edges in G, and use the v_i's to form a node set S^*. Let KPT be the mean of the expected prevention of S^* over the randomness in S^* and the prevention process. Due to the submodularity of the prevention function, it can be verified that KPT increases with k and

$$\frac{n}{m}\Big(EPT - \mathbb{E}[\omega(M_R)]\Big) \leq KPT \leq OPT_L \tag{8}$$

Additionally,

LEMMA 5. *Let R be a random RRC set and $\omega_\pi(R)$ be the subwidth of R. Define*

$$\kappa(R) = 1 - \left(1 - \frac{\omega_\pi(R)}{m}\right)^k \tag{9}$$

Then, $KPT = n \cdot \mathbb{E}[\kappa(R)]$, where the expectation is taken over the random choices of R.

Lemma 5 shows we can estimate KPT by first computing $n \cdot \kappa(R)$ on a set of random RRC sets and taking the average of the resulting measurements. However, as dictated by Chernoff bounds, if we want to obtain an estimate of KPT with $\delta \in (0, 1)$ relative error with at least $1 - n^{-l}$ probability then the number of samples required is $\Omega(ln \log n \cdot \delta^{-2}/KPT)$. That is, the number of measurements required to estimate KPT depends on KPT itself. This issue is also encountered in [24] and we are able to resolve it by mimicking their adaptive sampling approach, which dynamically adjusts the number of measurements based on the observed values of $\kappa(R)$, under the MCIC model.

Algorithm 3 KptEstimation($\mathcal{G}$,k,A_C)

```
1: for $i = 1$ to $\log_2 n - 1$ do
2:     Let $c_i = (6l \log n + 6 \log(\log_2 n)) \cdot 2^i$
3:     Let $sum = 0$
4:     for $j = 1$ to $c_i$ do
5:         Generate a random RRC set $R$
6:         $\kappa(R) = 1 - (1 - \frac{\omega_\pi(R)}{m})^k$
7:         $sum = sum + \kappa(R)$
8:     if $sum/c_i > 1/2^i$ then
9:         return $KPT^* = n \cdot sum/(2 \cdot c_i)$
10: return $KPT^* = 1$
```

Estimating KPT. Algorithm 3 presents the sampling approach for estimating KPT. The high level idea is as follows. Since KPT is an unknown quantity, we begin with the assumption that it takes on the value $n/2$. Then, we compute an estimate for the expected value of $\kappa(R)$ based on a relatively few number of samples. Chernoff bounds allow us to determine if the computed value of $KPT = n \cdot \kappa(R)$ is a good estimator and, if so, the algorithm terminates. However, if the estimate is much smaller than $n/2$ we apply the standard doubling approach and generate an increased number of samples to determine if KPT takes on a value close to half the initial estimate. The algorithm continues computing estimates for KPT, based on an increasing number of samples, and comparing to values that halve in size until the error bounds dictated by Chernoff bounds indicate we have reached a suitably precise estimation of KPT.

More specifically, in each iteration (Lines 2-7), the goal of Algorithm 3 is to compute the average value of $\kappa(R)$ over c_i randomly generated RRC sets from $\mathcal{G}$. As described in Lemma's 6 and 7 below, the c_i are chosen carefully such that if the average computed for $\kappa(R)$ over the c_i samples is greater than 2^{-i} then we can conclude that we have a good estimate for KPT with high probability. More precisely, that the expected value of $\kappa(R)$ is at least half of the average computed. Conversely, if the average computed is too small then Chernoff bounds imply that we have a bad estimate for KPT and the algorithm proceeds to the next iteration.

In the case that the true value of KPT is very small, the algorithm will terminate in the $\log_2 n$-th iteration and return $KPT^* = 1$, which equals the smallest possible KPT (since each node in the seed set can always save itself assuming $INF_C \geq k$). As we will show in the next section, $KPT^* \in [KPT/4, OPT_L]$ holds with a high probability. Thus, setting $\theta = \lambda/KPT^*$ ensures Algorithm 1 is correct and achieves the expected runtime complexity in Equation 7.

Performance Bounds. Proving the correctness and demonstrating bounds on the runtime for Algorithm 3 requires a careful analysis of the algorithm's behaviour. As shown below, we make use of two lemmas to prove that the algorithm's estimate of KPT^* is close to KPT.

Let $\mathcal{K}$ be the distribution of $\kappa(R)$ over random RRC sets in $\mathcal{G}$ with domain $[0,1]$. Let $\mu = KPT/n$, and s_i be the sum of c_i i.i.d. samples from $\mathcal{K}$, where c_i is defined as $c_i = (6l\log n + 6\log(\log_2 n)) \cdot 2^i$. Chernoff bounds give

LEMMA 6. *If $\mu \leq 2^{-j}$, then for any $i \in [1, j-1]$,*

$$Pr\left[\frac{s_i}{c_i} > \frac{1}{2^i}\right] < \frac{1}{n^l \cdot \log_2 n} \tag{10}$$

By Lemma 6, if $KPT \leq 2^{-j}$, then Algorithm 3 is very unlikely to terminate in any of the first $j-1$ iterations. This prevents the algorithm from outputting a KPT^* too much larger than KPT.

LEMMA 7. *If $\mu \geq 2^{-j}$, then for any $i \geq j+1$,*

$$Pr\left[\frac{s_i}{c_i} > \frac{1}{2^i}\right] > 1 - \left(\frac{1}{n^l \cdot \log_2 n}\right)^{2^{i-j-1}} \tag{11}$$

By Lemma 7, if $KPT \leq 2^{-j}$ and Algorithm 3 enters iteration $i > j+1$, then it will terminate in the i-th iteration with high probability. This ensures that the algorithm does not output a KPT^* that is too much smaller than KPT.

Based on Lemmas 6 and 7, we prove the following theorem for the correctness and expected runtime of Algorithm 3.

THEOREM 2. *When $n \geq 2$ and $l \geq 1/2$, Algorithm 3 returns $KPT^* \in [KPT/4, OPT_L]$ with at least $1 - n^{-l}$ probability, and runs in $O(l(m+n)(1/(1-\gamma))\log n)$ expected time. Furthermore, $\mathbb{E}[\frac{1}{KPT^*}] < \frac{12}{KPT}$.*

4.3 Improved Parameter Estimation

This section presents a heuristic for improving the practical performance of RPS, without affecting its asymptotic guarantees, by improving the estimated lower bound on OPT_L.

Algorithm 4 RefineKPT($\mathcal{G}$,k,A_C, KPT^*, ϵ')

```
1: Let $\lambda' = (2+\epsilon')ln\log n \cdot (\epsilon')^{-2}$.
2: Let $\theta' = \lambda'/KPT^*$.
3: Generate $\theta'$ random RRC sets; put them into a set $\mathcal{R}'$.
4: Initialize a node set $S'_k = \emptyset$.
5: for $i = 1$ to $k$ do
6:     Identify node $v_i$ that covers the most RRC sets in $\mathcal{R}'$.
7:     Add $v_i$ to $S'_k$.
8:     Remove from $\mathcal{R}'$ all RRC sets that are covered by $v_i$.
9: Let $f$ be the fraction of the original $\theta'$ RRC sets that
   are covered by $S'_k$.
10: Let $KPT' = f \cdot n/(1+\epsilon')$
11: return $KPT^+ = max\{KPT', KPT^*\}$
```

The KPT^* output by Algorithm 3 largely determines the efficiency of RPS. If KPT^* is close to OPT_L, then $\theta = \lambda/KPT^*$ is small and Algorithm 1 only needs to generate a relatively small number of RRC sets. However, if $KPT^* \ll OPT_L$ then the efficiency of Algorithm 1 degrades rapidly and, in turn, so does the overall performance of RPS.

To remedy this issue, we add an intermediate step before Algorithm 1 to refine KPT^* into a potentially tighter lower-bound of OPT_L. The intuition behind this heuristic is to generate a reduced number θ' of random RRC sets, placing them into a set $\mathcal{R}'$, and then apply the greedy approach (for the maximum coverage problem) on $\mathcal{R}'$ to obtain a good estimator for the maximum expected prevention in $\mathcal{R}'$. Thus, we can use the estimation as a good lower-bound for OPT_L.

Algorithm 4 shows the pseudo-code of the intermediate step. It first generates θ' random RRC sets and invokes the greedy approach for the maximum coverage problem on $\mathcal{R}'$ to obtain a size-k node set S'_k. Algorithm 4 computes the fraction f of RRC sets that are covered by S'_k so that, by Corollary 1, $f \cdot n$ is an unbiased estimation of $\mathbb{E}[\pi(S'_k)]$. We set θ' based on the KPT^* output by Algorithm 3 to a reasonably large number to ensure that $f \cdot n < (1+\epsilon') \cdot \mathbb{E}[\pi(S'_k)]$ occurs with at least $1 - n^{-l}$ probability. Based on this, Algorithm 4 computes $KPT' = f \cdot n/(1+\epsilon')$ scaling $f \cdot n$ down by a factor of $1+\epsilon'$ to ensure that $KPT' \leq \mathbb{E}[\pi(S'_k)] \leq OPT_L$. The final output of Algorithm 4 is $KPT^+ = max\{KPT', KPT^*\}$. Below we give a lemma that shows the theoretical guarantees of Algorithm 4.

LEMMA 8. *Given that $\mathbb{E}[\frac{1}{KPT^*}] < \frac{12}{KPT}$, Algorithm 4 runs in $O(l(m+n)(1/(1-\gamma))\log n/(\epsilon')^2)$ expected time. In addition, it returns $KPT^+ \in [KPT^*, OPT_L]$ with at least $1-n^{-l}$ probability, if $KPT^* \in [KPT/4, OPT_L]$.*

Summary of steps. In summary, we integrate Algorithm 4 into RPS and obtain an improved solution (referred to as RPS^+) as follows. Given $\mathcal{G}$, k, A_C, ϵ, and l, we first invoke Algorithm 3 to derive KPT^*. After that, we feed $\mathcal{G}$, k, A_C, KPT^*, and a parameter ϵ^* (as defined in [24]) to Algorithm 4, and obtain KPT^+ in return. Then, we compute $\theta = \lambda/KPT^+$, where λ is as defined in Equation 4. Finally, we run Algorithm 1 with $\mathcal{G}$, k, A_C, and θ as the input and get the output S^*_k as the final result of prevention maximization.

By Theorems 1 and 2, Equation 7, and the union bound, *RPS* runs in $O((k+l)(m+n)(1/(1-\gamma))\log n/\epsilon^2)$ expected time and it can be verified that when $\epsilon' \geq \epsilon/\sqrt{k}$, RPS^+ has the same time complexity as *RPS*. Furthermore, RPS^+ returns a $(1-1/e-\epsilon)$-approximate solution with at least $1-3n^{-l}$ probability and the success probability can be increased to $1-n^{-l}$ by scaling l up by a factor of $1+\log 3/\log n$.

5. LOWER BOUNDS

Comparison with *Greedy*. As mentioned previously, *Greedy* runs in $O(kmnr)$ time, where r is the number of Monte Carlo samples used to estimate the expected prevention of each node set. Budak et al. do not provide a formal result on how r should be set to achieve a $(1-1/e-\epsilon)$-approximation ratio in the MCIC model; instead, they only point out that when each estimation of expected prevention has ϵ related error, *Greedy* returns a $(1-1/e-\epsilon)$-approximate solution for a certain ϵ' [4]. In the following lemma, we present a more detailed characterization of the relationship between r and *Greedy*'s approximation ratio in the MCIC model:

LEMMA 9. Greedy *returns a* $(1-1/e-\epsilon)$*-approximate solution with at least* $1-n^{-l}$ *probability, if*

$$r \geq (8k^2+2k\epsilon)\cdot n\cdot\frac{(l+1)\log n+\log k}{\epsilon^2\cdot OPT_L} \qquad (12)$$

Assume that we know OPT_L in advance and set r to the smallest value satisfying the above inequality, in *Greedy*'s favour. In that case, the time complexity of *Greedy* is $O(k^3lmn^2\epsilon^{-2}\log n/OPT_L)$. Towards comparing *Greedy* to RPS, we show the following upper bound on the value of γ:

CLAIM 1. $\gamma \leq \frac{n}{n+1}$

Claim 1 shows that the expected runtime for *RPS* is at most $O((k+l)mn\epsilon^{-2}\log n)$. As a consequence, given that $OPT_L \leq n$, the expected runtime of *Greedy* is always more than the expected runtime of *RPS*. In practice, we observe that for typical social networks $OPT_L \ll n$ and $\frac{1}{1-\gamma} \ll n+1$ resulting in superior scalability of *RPS* compared to *Greedy*. Table 3 confirms that $\frac{1}{1-\gamma} \ll n+1$ on our small datasets.

A Lower Bound for EIL. In the theorem below, we provide a lower bound on the time it takes for any algorithm to compute a β-approximation for the EIL problem given uniform node sampling and an adjacency list representation. Thus, we rule out the possibility of a sublinear time algorithm for the EIL problem for an arbitrary β.

THEOREM 3. *Let* $0<\epsilon<\frac{1}{10e}$, $\beta \leq 1$ *be given. Any randomized algorithm for EIL that returns a set of seed nodes with approximation ratio* β*, with probability at least* $1-\frac{1}{e}-\epsilon$*, must have a runtime of at least* $\frac{\beta(m+n)}{24\min\{k,1/\beta\}}$.

The proof of Theorem 3 uses Yao's Minmax Lemma for the performance of Las Vegas (LV) randomized algorithms on a family of inputs [26]. Precisely, the lemma states that the least expected cost of deterministic LV algorithms on a distribution over a family of inputs is a lower bound on the expected cost of the optimal randomized LV algorithm over that family of inputs. We build such an input family of lower bound graphs via the use of a novel gadget.

Name	$\lvert V\rvert$	$\lvert E\rvert$	Average degree
nethept	15,229	62,752	4.1
word_assoc	10,617	72,172	6.8
dblp-2010	326,186	1,615,400	6.1
cnr-2000	325,557	3,216,152	9.9
ljournal-2008	5,363,260	79,023,142	28.5

Table 2: Dataset Statistics

6. EXPERIMENTS

In this section, we present our experimental results. All of our algorithms are implemented in C++ (available at `https://github.com/stamps`) and tested on a machine with dual 6 core 2.10GHz Intel Xeon CPUs, 128GB RAM and running Ubuntu 14.04.2.

Datasets. The network statistics for all of the datasets we consider are shown in Table 2. We obtained the datasets from Laboratory of Web Algorithmics.[2] We divide the datasets by horizontal lines according to their size, small (S), medium (M), and large (L).

Propagation Model. We consider the MCIC model (see Section 2.1) of Budak et al. We set the propagation probability of each edge e as follows: we first identify the node v that e points to, and then set $p(e)=1/i$, where i denotes the in-degree of v. This setting of $p(e)$ is widely adopted in prior work [6, 5, 13, 25].

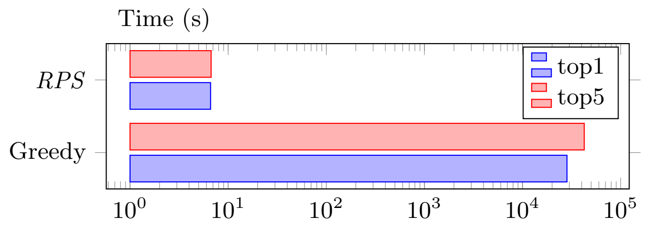
Parameters. Unless otherwise specified, we set $\epsilon = 0.1$ and $k = 10$ in our experiments. We set l in a way that ensures a success probability of $1-1/n$. For *Greedy*, we set the number of Monte Carlo steps to $r = 10000$, following the standard practice in the literature. Note that this choice of r is to the advantage of *Greedy* because the value of r required to achieve the same theoretical guarantees as *RPS* in our experiments is always much larger than 10000. In each of our experiments, we repeat each run three times and report the average result.

We are interested in simulating the misinformation prevention process when the bad campaign C has a sizable influence on the network to best demonstrate how the techniques could be used in real world settings. That is, we believe the scenario in which we are attempting to prevent the spread of misinformation when the bad campaign has the ability to influence a large fraction of the network to be more relevant than when only a very small number of users would adopt the bad campaign. Towards this end, we first compute the *top-k* influential vertices for each network and then randomly select the seed set A_C from the *top-1* and *top-5* vertices for each experiment. This process ensures the misinformation has a large potential influence in the network.

The measures of effectiveness in our experiments are the *algorithm efficiency* measured in runtime and the *algorithm accuracy* measured in percentage of nodes saved. Our goal is to demonstrate the superior runtime of *RPS* as the accuracy has been covered extensively in [4].

Running-time Results. First, we plot the runtimes of *Greedy* and *RPS* for a single seed in Figure 3 and observe that *RPS* provides a significant improvement of several orders of magnitude over *Greedy*. Note, we only compare *RPS* to *Greedy* on the smallest networks due to the substantial

[2]`http://law.di.unimi.it/datasets.php`



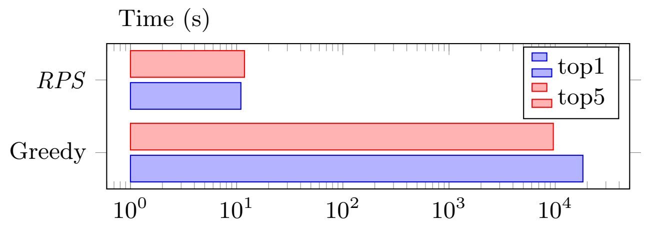


Figure 3: Runtimes comparison between *RPS* and *Greedy* for wordassociation-2011 (left) and nethept (right) datasets.

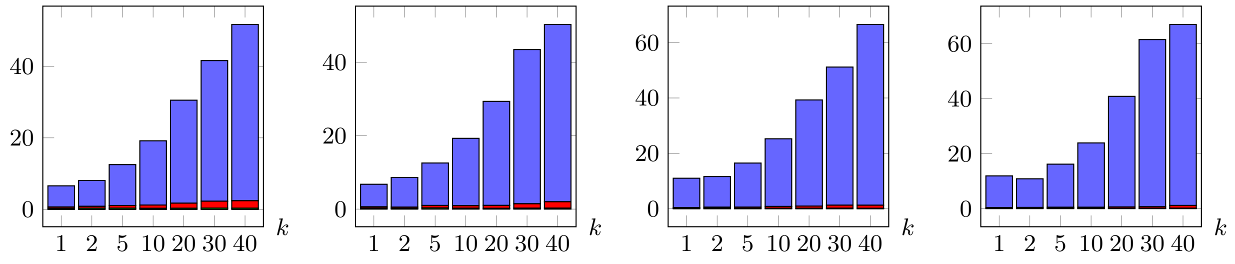


Figure 4: Breakdown of computation time (s) for small datasets. Blue stack corresponds to Algorithm 1, red to Algorithm 4, and green (which is almost invisible) to Algorithm 3. Listed left to right: word_assoc top1, word_assoc top5, nethept top1, nethept top5.

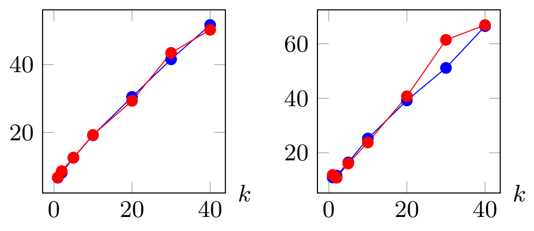


Figure 5: Runtimes (s) for small datasets. word_assoc on the left and nethept on the right with blue for top1 and red for top5.

	word_assoc		nethept	
k	top1	top5	top1	top5
1	23.4471	25.9804	48.3194	48.619
10	24.8521	26.6518	60.5	43.1875
20	24.7509	25.2607	57.0111	61.4167
max	10,618	10,618	15,230	15,230

Table 3: $\frac{1}{1-\gamma}$ values for small datasets.

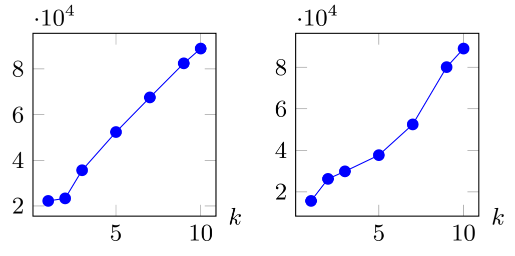


Figure 6: Runtimes (s) for ljournal-2008. Left is top1 and right is top5.

runtime required for *Greedy*. Furthermore, for similar scaling issues of *Greedy*, we restrict our comparison to $k = 1$. However, since both approaches scale linearly with k we can conclude that *RPS* offers a tremendous runtime improvement over the approach of [4].

Next, we show the total runtimes (Figures 5, 9), computation breakdowns (Figures 4, 7, 10), and prevention (Figures 8, 11) for each dataset. We observe that the vast majority of the computation time is spent on generating the RRC sets for $\mathcal{R}$. Furthermore, we see that, consistent with the results reported in [4], *RPS* quickly approaches a maximal expected prevention value as k increases across all datasets.

We now compare the runtime trends of our results for the EIL problem to those of [24] for the IM problem. Tang et al. report that, when k increases, the runtime of their approach (*TIM*) tends to decrease before eventually increasing. They explain this by considering the breakdown of the computation times required by each algorithm in *TIM*. They observe that the computation time is mainly incurred by their analog to Algorithm 1 (the node selection phase) which is primarily determined by the number θ of RR sets that need to be generated. They have $\theta = \lambda/KPT^+$, where λ is analogous to ours, and KPT^+ is a lower-bound on the optimal influence of a size-k node set. In both the IC and MCIC models, the analogs of λ and KPT^+ increase with k, and it happens that for the IM problem, Tang et al. observe that the increase of KPT^+ is more pronounced than that of λ for smaller values of k, which leads to the decrease in *TIM*s runtime.

On the contrary, for the EIL problem, the increase of KPT^+ does not dominate to a point that the runtime of *RPS* decreases as k increases. Instead, we see a linear increase in runtime as k increases for all the networks considered (Figures 5, 9). To explain, consider how KPT^+ grows in each setting. In the MCIC model we see that KPT^+ rapidly approaches its maximal value which corresponds to the growth of KPT^+ plateauing much sooner. In contrast, in the IC model, the analogous KPT^+ value continues to grow at a significant rate for a wider range of k values since the ceiling for the maximal influence is not tied to a second campaign, as it is in the MCIC model. As such, the influence estimates do not level off as quickly. This translates to the growth of KPT^+ outpacing the growth of λ.

Memory Consumption. Our final set of experiments

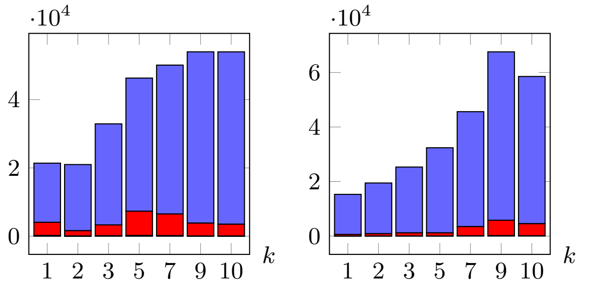


Figure 7: Breakdown of computation time (s) for ljournal-2008. Blue stack corresponds to Algorithm 1, red to Algorithm 4, and green to Algorithm 3. Left is top1 and right is top5.

monitors the memory consumption required to store the RRC set structure $\mathcal{R}$. We observe that the size of $\mathcal{R}$ for the EIL problem is larger than that required by the IM problem. In the "hypergraph" nomenclature due to Borgs et al. $\mathcal{R}$ is viewed as a hypergraph with each RRC set in $\mathcal{R}$ corresponding to a hyperedge. We observe that the hyperedges generated for the IM problem are non-empty in every iteration of the algorithm. Additionally, each hyperedge has relatively small size. The result is that the hypergraph generated for the IM problem is very dense, but each hyperedge is relatively "light" (i.e. it contains few nodes).

In contrast, in each iteration of *RPS* we have a substantial probability to produce an empty RRC set, since we require that a randomly selected node is in the randomized BFS tree resulting from the influence propagation process initialized at A_C. Even though our seed sets for campaign C are chosen at random from the *top-k* sets, and thus are highly influential (resulting in a large randomized BFS tree), the probability of overlap with a randomly selected node is still small due to the vast size of the networks considered. These empty RRC sets are necessary for the computation of the expected prevention to be accurate, but results in a hypergraph that differs significantly from that of the IM problem.

In particular, since the *generateRRC* algorithm is a deterministic BFS (with specialized stopping conditions to account for cutoff nodes) it reaches a much larger fraction of the network. Therefore, while there are far fewer non-empty hyperedges generated, they are much large in size: often on the order of half the network. Thus, the resulting hypergraph is sparse, but contains very "heavy" edges.

These two opposing metrics, a dense hypergraph with "light" hyperedges versus a sparse hypergraph with "heavy" hyperedges, result in the latter requiring more memory to store. Memory consumption statistics are shown in Fig. 12. Despite a larger memory requirement compared to the single campaign setting we show that our approach has the ability to scale far beyond what was achieved by Budak et al. and provides orders of magnitude improvement for the runtime.

7. RELATED WORK

There exists a large body of work on the Influence Maximization problem first proposed by Kempe et al. [15]. The primary focus of the research community has been related to improving the practical efficiency of the *Greedy* approach. These works fall into two categories: heuristics that trade efficiency for approximation guarantees [13, 25] and practical optimizations that speed up *Greedy* while retaining the approximation guarantees [16, 6, 9]. Leskovec et al. [16] present algorithmic optimizations to *Greedy* that reduces the runtime up to 700-fold by avoiding the evaluation of the expected influence for many node sets, while [6, 9] provide further refinements to the approach of [16]. Despite these advancements, the ability of *Greedy* to scale to web-scale networks remained unfeasible.

The breakthrough work of Borgs' et al. [3] brought the first asymptotic runtime improvements while maintaining the $(1-1/e-\epsilon)$-approximation guarantees with their *reverse influence sampling* technique. Furthermore, they prove their approach is near-optimal under the IC model. Tang et al. [24] presented practical and theoretical improvements to the approach of Borgs' et al. that lead to the elimination of a large hidden constant in the runtime and novel heuristics that result in up to 100-fold improvements to the runtime of their *TIM* approach.

Related to the ongoing work in identifying influential users in a network, there has been a significant amount of work on incorporating the spread of competing campaigns [2, 18, 22, 12] and the spread of misinformation and rumours [4, 7, 20, 17] by augmenting or extending the IC model. The work most related to ours is [18] where they extend the *reverse influence sampling* technique of [3] to competing campaigns (such as two competing products). However, their work differs from ours in three important ways: first, they use the COIC model of [4] (as opposed to the MCIC model) where the edge probabilities are campaign oblivious. That is, no matter which information reaches a node, it is propagated with a single fixed probability for both campaigns. Budak et al. note that this alternative model does not capture the notion of misinformation as well as the MCIC model, but instead that it is better suited for the spread of multiple competing campaigns. More importantly, the COIC model does not require the concept of *blocking* defined in this paper. Since all probabilities are the same, shortest paths suffice for determining which campaign a node adopts. Therefore the work of [18] does not require the new definitions we introduce to close the shortest paths gap in the analysis of [4] for the MCIC model. Secondly, since they operate in the COIC model, they are studying a fundamentally different problem. They are studying the influence maximization problem between competing products and not misinformation prevention. Finally, the scalability of their approach is orders of magnitude less than ours as their largest experiments are on a network with only 500,000 edges.

8. CONCLUSION

In this work we presented *RPS*, a novel and scalable approach to the EIL problem. We gave proofs of correctness and a detailed running-time analysis of our approach. Furthermore, we experimentally verified the performance of our algorithm on a collection of large social networks and observed a significant improvement over the state-of-the-art *Greedy* approach. Finally, we provided additional theoretical results in the form of two lower bounds: one on the running-time requirement for any approach to solve the EIL problem and another of the number of Monte Carlo simulations required by *Greedy* to return a correct solution with high probability.

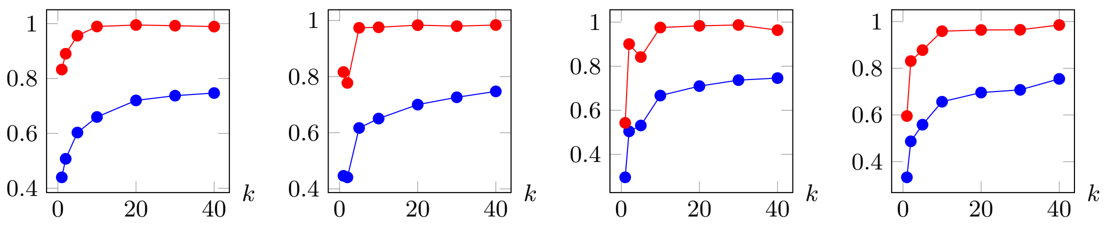


Figure 8: Expected prevention (%) for small datasets. Blue plot corresponds to KPT^+ and red plot to RPS. Listed left to right: word_assoc top1, word_assoc top5, nethept top1, nethept top5.

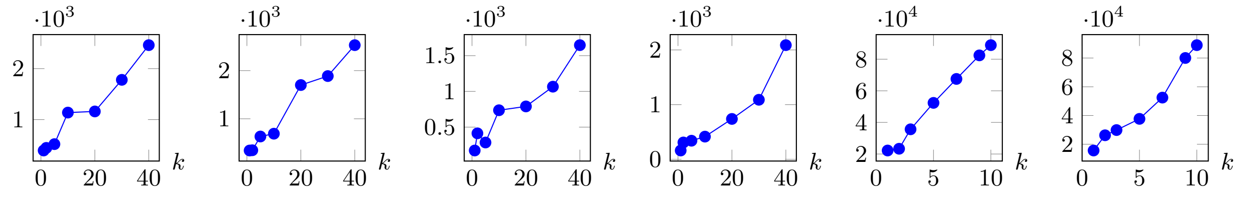


Figure 9: Runtimes (s) for medium & large datasets. Listed left to right: dblp top1, dblp top5, cnr top1, cnr top5, ljournal-2008 top1, ljournal-2008 top5.

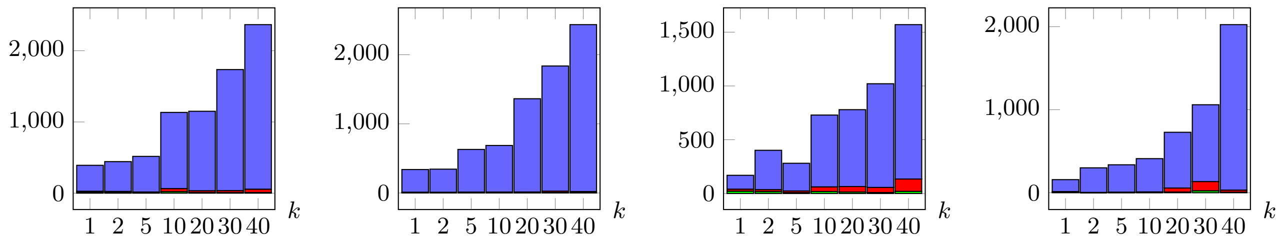


Figure 10: Breakdown of computation time (s) for medium datasets. Blue stack corresponds to Algorithm 1, red to Algorithm 4, and green to Algorithm 3. Listed left to right: dblp top1, dblp top5, cnr top1, cnr top5.

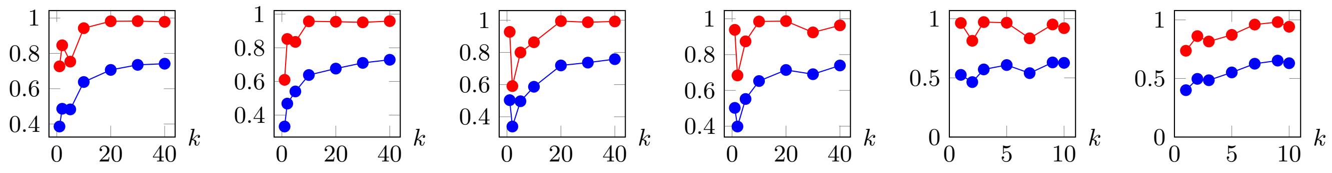


Figure 11: Expected prevention (%) for medium & large datasets. Blue plot corresponds to KPT^+ and red plot to RPS. Listed left to right: dblp top1, dblp top5, cnr top1, cnr top5, ljournal-2008 top1, ljournal-2008 top5.

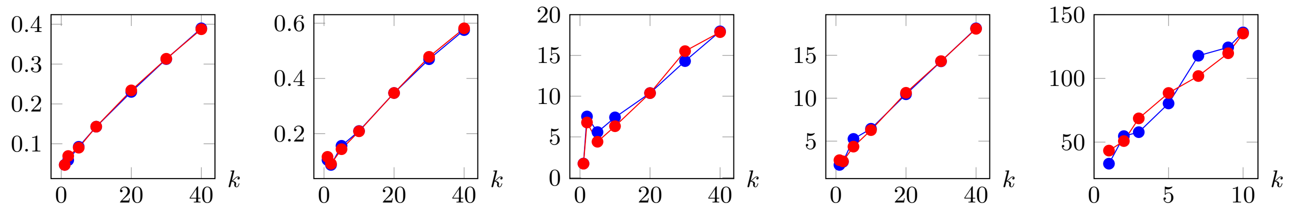


Figure 12: Memory consumption (Gb) for all datasets. Blue corresponds to top1 and red to top5. Listed left to right: word_assoc, nethept, cnr, dblp & ljournal-2008.

9. REFERENCES

[1] B. Abeshouse. Troll factories, bots and fake news: Inside the wild west of social media, February 2018.

[2] S. Bharathi, D. Kempe, and M. Salek. Competitive influence maximization in social networks. In *WINE'07*, pages 306–311, Berlin, Heidelberg, 2007. Springer-Verlag.

[3] C. Borgs, M. Brautbar, J. T. Chayes, and B. Lucier. Influence maximization in social networks: Towards an optimal algorithmic solution. *CoRR*, abs/1212.0884, 2012.

[4] C. Budak, D. Agrawal, and A. El Abbadi. Limiting the spread of misinformation in social networks. In *WWW'11*, 2011.

[5] W. Chen, Y. Wang, and S. Yang. Efficient influence maximization in social networks. In *KDD '09*, pages 199–208, New York, NY, USA, 2009. ACM.

[6] W. Chen, Y. Yuan, and L. Zhang. Scalable influence maximization in social networks under the linear threshold model. In *ICDM'10*, 2010.

[7] L. Fan, Z. Lu, W. Wu, B. Thuraisingham, H. Ma, and Y. Bi. Least cost rumor blocking in social networks. In *Distributed Computing Systems (ICDCS), 2013 IEEE 33rd International Conference on*, pages 540–549. IEEE, 2013.

[8] P. Foster. 'bogus' ap tweet about explosion at the white house wipes billions off us markets, April 2018.

[9] A. Goyal, F. Bonchi, L. V. S. Lakshmanan, and S. Venkatasubramanian. On minimizing budget and time in influence propagation over social networks. *Social Netw. Analys. Mining*, 3(2):179–192, 2013.

[10] C. Graham. Youtube employee's twitter account hacked to spread fake news during attack, April 2018.

[11] L. Hautala. Reddit was a misinformation hotspot in 2016 election, study says, December 2018.

[12] X. He, G. Song, W. Chen, and Q. Jiang. *Influence Blocking Maximization in Social Networks under the Competitive Linear Threshold Model*, pages 463–474.

[13] K. Jung, W. Heo, and W. Chen. Irie: Scalable and robust influence maximization in social networks. In *Data Mining (ICDM), 2012 IEEE 12th International Conference on*, pages 918–923. IEEE, 2012.

[14] D. Kempe. Structure and dynamics of information in networks. 2011.

[15] D. Kempe, J. Kleinberg, and E. Tardos. Maximizing the spread of influence through a social network. In *KDD'03*, 2003.

[16] J. Leskovec, D. Huttenlocher, and J. Kleinberg. Predicting positive and negative links in online social networks. In *WWW '10*, pages 641–650, New York, NY, USA, 2010. ACM.

[17] Y. Li, W. Chen, Y. Wang, and Z.-L. Zhang. Influence diffusion dynamics and influence maximization in social networks with friend and foe relationships. In *Proceedings of the Sixth ACM International Conference on Web Search and Data Mining*, WSDM '13, pages 657–666, New York, NY, USA, 2013. ACM.

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

[19] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher. An analysis of approximations for maximizing submodular set functions—i. *Mathematical Programming*, 14(1):265–294.

[20] N. P. Nguyen, G. Yan, M. T. Thai, and S. Eidenbenz. Containment of misinformation spread in online social networks. In *Proceedings of the 4th Annual ACM Web Science Conference*, WebSci '12, pages 213–222, New York, NY, USA, 2012. ACM.

[21] M. Oppenheim. Youtube shooting: Twitter and facebook explodes with misinformation and hoaxes, April 2018.

[22] N. Pathak, A. Banerjee, and J. Srivastava. A generalized linear threshold model for multiple cascades. In *Data Mining (ICDM), 2010 IEEE 10th International Conference on*, pages 965–970. IEEE, 2010.

[23] O. Solon. Facebook's failure: did fake news and polarized politics get trump elected?, November 2018.

[24] Y. Tang, X. Xiao, and Y. Shi. Influence maximization: Near-optimal time complexity meets practical efficiency. In *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, SIGMOD '14, pages 75–86, New York, NY, USA, 2014. ACM.

[25] C. Wang, W. Chen, and Y. Wang. Scalable influence maximization for independent cascade model in large-scale social networks. *Data Mining and Knowledge Discovery*, 25(3):545–576, 2012.

[26] A. C.-C. Yao. Probabilistic computations: Toward a unified measure of complexity. In *Foundations of Computer Science, 1977., 18th Annual Symposium on*, pages 222–227. IEEE, 1977.