Harvester

Influence Optimization in Symmetric Interaction Networks

Ivanov, Sergei; Karras, Panagiotis

Published in:
2016 IEEE International Conference on Data Science and Advanced Analytics (DSAA)

DOI (link to publication from Publisher):
10.1109/DSAA.2016.95

Creative Commons License
Unspecified

Publication date:
2016

Document Version
Accepted author manuscript, peer reviewed version

Harvester: Influence Optimization in Symmetric Interaction Networks

***Abstract*—The problem of optimizing influence diffusion in a network has applications in areas such as marketing, disease control, social media analytics, and more. In all cases, an initial set of influencers are chosen so as to optimize influence propagation. While a lot of research has been devoted to the influence maximization problem, most solutions proposed to date apply on directed networks, considering the undirected case to be solvable as a special case. In this paper, we propose a novel algorithm, Harvester, that achieves results of higher quality than the state of the art on symmetric interaction networks, leveraging the particular characteristics of such networks. Harvester is based on the aggregation of instances of live-edge graphs, from which we compute the influence potential of each node. We show that this technique can be applied for both influence maximization under a known seed size and also for the dual problem of seed minimization under a target influence spread. Our experimental study with real data sets demonstrates that: (a) Harvester outperforms the state-of-the-art method, IMM, in terms of both influence spread and seed size; and (b) its variant for the seed minimization problem yields good seed size estimates, reducing the number of required trial influence spread estimations by a factor of two; and (c) it is scalable with growing graph size and robust to variant edge influence probabilities.**

I. Introduction

The enormous growth of information represented in the form of networks, such as protein interaction networks, online social networks, communication networks, etc. has created the opportunity to study the spread of artifacts, information, diseases, trends, as well as business opportunities leveraging the results of such studies; moreover, as large-scale networks with users' relationships and history of online activity become available for researchers, they pose new computational challenges. However, despite a widespread belief that such networks are valuable in biotechnology, disease control, advertising, and marketing, much of this potential remains untapped. For example, STRING, a database of known and predicted protein interactions, currently covers 9,643,763 proteins from 2,031 organisms[1]; microblogging social service Twitter was valued at $35 billion[2] in January 2014, but its expected revenue for the same year was around $1.2 billion [3]. In effect, new algorithms that can efficiently deal with massive-scale networks are of paramount interest.

A problem that has received a great amount of attention in this regard is the *influence maximization problem* [1]. Consider the following scenario: a mobile development company releases a new game, which is promoted through advertising in social networks. The company plans to target a few individuals promising free subscription who will then share a short description of the game with their friends. In turn, their friends may find the game interesting and spread the information further to the circle of their friends, and so on. This word-of-mouth effect, in which people adopt new products because their friends did so already, is called *viral marketing*.

Realistically, a company has a budget constraint on the number of users it can target at the outset. In the influence maximization problem, we aim to maximize the expected *spread* (or adoption) of a product, trend, or behavior, in the network by selecting a *seed set* of initial adopters of a fixed size. The propagation of influence in the network may be carried out by some diffusion model over a network where each edge is assigned a probability that influence is carried across it, such as the Independent Cascade or Linear Threshold model. Past research has proposed solutions to this problem [2], [3], [4], [5], [6], showing that a good initial choice of nodes can result in manyfold increase in probabilistically expected influence spread compared to choosing nodes uniformly at random.

While the influence maximization problem has attracted abundant attention, extant solutions consider by default the influence propagates on a *directed* graph. While this is a general scenario, it remains the case that many real-world interaction networks, such as some neural networks, protein interaction networks, and communication networks, represent *symmetric interactions*, and are therefore undirected, i.e., links are symmetric; given such symmetric links, influence is also exercised symmetrically across a connection. Nevertheless, other than a theoretical analysis in [7], previous research has not attempted to provide algorithms tailored for undirected graphs in particular; they provide solutions for directed graphs, and assume that the undirected case can be handled as a special case, without need for special solutions.

Furthermore, to the best of our knowledge, most algorithms proposed in previous research share a fundamental characteristic: they build their solutions in a monotonic, incremental, greedy manner, adding one node at a time to the output seed set. Nevertheless, even though this strategy allows for the provision of approximation guarantees due to the submodularity of the underlying objective function [1], it remains the case that the optimal solution to the problem does not share such monotonicity - after all, if it did, the problem would not be NP-hard: the optimal seed set of size k is *not necessarily* a subset of the optimal seed set of size $k+1$; yet, extant solutions work as if that were the case; therefore, there is arguably work room for algorithm that would work in a different manner, returning substantially different solutions for different seed set sizes.

Last, the dual variant of the problem, which is of no less importance, has not received much attention either. In this

[1] http://www.wolframalpha.com/input/?i=Market+capitalization+of+Twitter+in+January+2014

[2] Wolfram Alpha: Market capitalization of Twitter in January 2014

[3] TechCrunch: Twitter Beats In Q1 With $250M In Revenue And Picks Up 14M New Monthly Active Users

seed minimization problem, the objective is to minimize the number of initial adopters that can achieve a certain critical level of influence spread in the network. Following up on the previous example, a mobile development company may want to reach a critical number of adopters so that a game becomes a hot app that can be promoted by reviewers of media websites, special sections in game stores, and social network celebrities. Once such a critical level of adoption is reached, it will eventually bring about even bigger cascade of adoption and increased sales for the company. The seed minimization problem corresponds to such a scenario, where we are searching for a minimal number of selected users who can bring influence propagation to a certain level; their minimal number ensures minimization of initial costs. Solutions for this problem variation have been proposed in the recent years to address some of the challenges [8], [9], [10]. However, questions of efficiency and scalability remain unanswered.

In this paper, we address each of the aforementioned three challenges: we provide novel influence maximization algorithms especially tailored for undirected graphs, leveraging the characteristics of such graphs; our solutions are constructed in a *non-monotonic* manner, producing substantially different output seed sets for different seed sizes k; last, we address the dual, seed minimization problem in a non-trivial yet efficient manner, and not as a by-product of an algorithm for the primal influence maximization problem. In both problem cases, our algorithms are based on *instance aggregation*, where an instance corresponds to a live-edge graph, i.e., a graph extracted by letting each possible edge be either *active* or *blocked*, according to its activation probability. Our algorithms consist of two phases; the first, *accumulation phase*, assigns scores to all nodes in the graph based on the size of a connected component they belong to in each live-edge instance, which captures a node's potential to propagate influence on that instance; the second, *penalization phase*, selects nodes ordered by accumulated score, while penalizing their immediate neighbors. We name our heuristic *Harvester*, drawing from the analogy of sowing seeds and mowing harvest; such an analogy applies to the problem's objective, in which a seed results into an influence spread as a harvest, and also to the way our algorithm operates, as the collected scores can be viewed as the seed sowed and the solution as the harvest; this harvested solution is not monotonic with respected to seed size k, as the k parameter is taken into account and plays a role throughout the operation of our algorithm.

Our approach works for both the influence maximization problem and its dual, seed minimization problem. Its application on the former is unproblematic, as the number of nodes that have to be selected is a constraint of the problem known in advance. However, its application on the latter problem is more challenging, just as it is for any influence maximization algorithm. In principle, any such algorithm can be used for solving the seed minimization problem via a trial-and-error process, using, e.g., binary search on the domain of seed sizes, until convergence to the minimum seed size that achieves the desired spread coverage. However, this trial-and-error process is bound to be very costly, because every step thereof requires an expensive evaluation of influence spread, and the number of steps may be high.

Surprisingly, our instance aggregation method has a prime advantage over competing methods when applied on the seed minimization problem: it yields a good *a priori* estimate of the final seed size, thereby reducing the number of required influence spread estimations. Such an estimate is achieved using collected statistics, in particular the average number, over all live-edge graph instances, of connected components who cumulatively achieve the target spread. Using this estimate, as we show, we need less than half the number of influence spread evaluations carried out during the search process, in comparison to other heuristics.

The rest of the paper is organized as follows. In Section II we present related work in the area. Sections III and IV outline our algorithm for influence maximization and its adaptation for seed minimization. We present our experimental results in Section V and conclude the paper in Section VI.

II. Related Work

The influence maximization problem has attracted a lot of attention in the last decade; Kempe et al. [1] first posed it for two classic diffusion models, Independent Cascade (IC) and Linear Threshold (LT), and provided greedy algorithms with approximation guarantees, relying on iterative re-computations of the expected marginal influence spread gain for each node. Chen et al. [3], [11] proved that computing the expected influence spread for any given seed set is $\#\mathbf{P}$-hard, prompting the study of fast scalable algorithms. Cheng et al. [12] provide a static implementation of the greedy algorithm that trades off repetitive Monte Carlo simulations for higher memory consumption.

Leskovec et al. [13] proposed *lazy evaluations*, a technique that reduces the number of influence spread evaluations in the greedy algorithm. The idea is that if, at any iteration, the marginal gain of a node u exceeds the marginal gain of another node v at a previous iteration, then we do not need to recompute the influence spread for v, due to the influence functions's *submodularity* (diminishing returns property). Goyal et al. [14] proposed CELF++, an improvement over lazy evaluations that gains about 1000-fold speed-up over the original greedy approach.

Another approach for tackling the running time of the greedy algorithm is to eschew independent influence computations for each node, and perform batch estimates instead [15], [2]. More drastic ways to reduce runtime attempt to approximate influence spread [2], [3], [11], [4], [16], [5], [6]. For instance, Chen et al. [3] propose leveraging arborescences, local structures for which we can estimate influence spread in polynomial time; Kim et al. [17] suggest the Independent Path Algorithm (IPA), for the Independent Cascade diffusion model, which takes a heuristic shortcut to approximating influence by considering an independent influence path as an influence evaluation unit; most recently, Tang et al. [6] apply a martingale-based approach to achieve high-quality solutions in near-linear time, which provides the state of the art in that respect. Yet the aforementioned algorithms are tailored for *directed networks*, while most of them, with the cardinal exception of [5], [6], follow the logic of the greedy algorithm [1], forfeiting the opportunity to produce non-monotonic solutions; a recent heuristic [18] exploits this very greedy logic, by finding and leveraging a ranking of nodes self-consistent with their ranking-based marginal influence spreads.

Alternatively, Wang et al. [19] follow a community-based approach to the problem; Jiang et al. [20] apply simulated annealing on it; Chen et al. [21] study an extension that incorporates the emergence and propagation of negative opinions. Goyal et al. [22] examine how available historical propagation traces can be leveraged to learn how influence actually flows in the network and uses this to estimate expected influence spread. Li et al. [23] examine the problem in a conformity-aware manner, taking into consideration not only people's ability to influence others, but also their inclination to be influence by their environment.

Less attention has been paid to the dual problem of influence maximization, the *seed minimization problem*: given a coverage threshold T, find a seed set of minimal size, such that it propagates influence to coverage T under a diffusion process. This problem was first considered by Ning Chen [8] under the *fixed threshold model*, a variation of the LT model, in which a deterministic activation threshold is chosen for each node; Chen shows that the problem is inapproximable within a poly-logarithmic factor unless **NP** $\subseteq$ DTIME($n^{polylog(n)}$) by reduction to the Minimum Representative problem, and provides a polynomial-time algorithm for the case the underlying graph is a tree. Ben-Zwi et al. [24] solved the problem for bounded tree-width graphs under the same model in $O(n^{O(w)})$, where w is the width of the graph. Goldberg and Liu [25] provided an LP-based approximation algorithm for the LT model with a deterministic threshold and the distinction that not only the active neighbors of a node v can influence v, but all active nodes that can reach v through other active nodes. Both aforementioned models are deterministic and not submodular.

Goyal et al. [9] develop a greedy algorithm that aims to minimize target set cost $c(S) = \sum_{v \in S} c(v)$, where $c(v) \geq 0$ are non-uniform node costs, a generalization of the seed minimization problem; they provide a bicriteria approximation, where, given a desired coverage η and a shortfall parameter $\epsilon > 0$, for the solution S it holds that $\sigma(S) \geq \eta - \epsilon$ and $c(S) \geq c(S^*)(1 + \ln(\eta/\epsilon)$, where $\sigma(S)$ is the spread achieved by set S and S^* the optimal solution, i.e., the offered solution exceeds the optimal one in terms of cost by a logarithmic factor, while its coverage falls short of the required coverage by ϵ. Zhang et al. [10] study the seed selection problem with a probabilistic guarantee, where the produced solution S should reach coverage η with probability at least P. They show that the set functions are not submodular in this case, while the problem is #**P**-hard, and provide a greedy algorithm with a loose approximation guarantee. More recently, Lei et al. [26] study the problem of Online Influence Maximization (OIM), where influence probability information is not given in advance; the solution relies on real-world user feedback to update its influence probability model while running influence campaigns. Any existing IM algorithm can be used in this framework [26].

III. Influence Maximization

In this section we present our heuristic for influence maximization under the IC model on undirected graphs.

A. Problem definition

Consider a social network as an undirected graph $G = (V, E)$, where V is a set of vertices that represent individuals and E is a set of edges that represent mutual relationship between individuals. For example, in the peer-to-peer network Gnutella,[4] nodes are hosts in the network and edges are connections among those hosts.

Each node $u \in V$ can be either *active* or *inactive*, depending on whether it has been already influenced or not. Once a node becomes active, it stays in that state until the end of the diffusion process. Each edge (u, v) is associated with a *propagation probability*, p_{uv}, i.e., the probability that active node u activates inactive node v, or vice versa.

We consider a popular propagation model, the Independent Cascade (IC) model. The IC propagation model defines a diffusion process that starts from the set of active nodes S_0 and goes through a sequence of steps as follows. Let S_i be a set of nodes in G that are activated during time step i. Then, at step $i+1$, any inactive node v with at least one neighbor in S_i can be activated with probability $1 - \prod_{i=1}^{w} (1 - p_{e_i})$, where p_{e_i} is the activation probability on edge e_i between v and a node $u \in S$, $1 \leq i \leq w$ and w the total number of such edges, allowing duplicate edges between v and the same node $u \in S$; hence, each edge is given exactly one chance to be activated, and the diffusion process stops when there are no newly activated nodes at a step j, i.e. $S_j = \emptyset$.

The diffusion process has to start from an initial *seed set* of active nodes S_0. The objective of the *influence maximization problem* for a parameter k is to find a target seed set S_0 of size k that maximizes the *expected* number of influenced nodes at the end of a diffusion process under the IC model, $\sigma(S_0)$. This problem is NP-hard [1].

B. The Live-Edge Graph

In the IC model, each node gets only a single chance to activate its neighbors after it is itself activated. However, it has been shown that the final set of active nodes can be equivalently found by means of a *live-edge graph* [1].

Consider a time step t of the IC diffusion process. A node u that has just become active, is then given a chance to activate a neighbor v along edge e_{uv} with probability $p_{e_{uv}}$. Such an activation of v by u is independent of other nodes in the network. Thus, edge e_{uv} is present in the network, or *live*, with probability $p_{e_{uv}}$, or, otherwise, it is *blocked*. Then the following proposition holds [1]:

Proposition 3.1: A node v is active iff there is a path from the set of initially activated nodes S_0 to v made entirely of live edges.

By Proposition 3.1, we can view the IC diffusion process as follows: we first decide whether an edge is live or blocked, and then, starting from seed set S_0, we activate all nodes reachable by other active nodes via live-edge paths. Let $R_G(S)$ be the set of nodes that is reachable from S in graph G and let $G' = (V', E')$, where $V' \equiv V$ and E' is the set of live edges in E, i.e. G' is the graph that results by keeping only live edges in G. Then, the final active set is the set of nodes reachable from

[4] http://snap.stanford.edu/data/p2p-Gnutella09.html

S_0 in G', $R_{G'}(S_0)$. We suggest that this alternative view of the diffusion process can be leveraged to measure the potential each node has to activate other nodes, thereby suggesting good choices for seed set S_0.

C. Direct Harvester

Our algorithm starts out from the following observation, which is specific to undirected networks: when we remove blocked edges from a connected undirected graph G, we end up with a live-edge graph instance G' having, in general, many disjoint connected components (CCs). Then, for a given seed budget of k nodes, we can *retrospectively* maximize the influence spread on instance G' *itself* by including in seed set S one node from each of the top-k CCs by size (breaking ties arbitrarily). That is so because, if we arbitrarily pick up a node v from a connected component CC and include it in S, then all other nodes in CC will be activated by the diffusion process, according to Proposition 3.1. Thus, by choosing any node from each of the k largest CCs, we ensure activating all nodes in those CCs, and hence maximizing influence spread in graph instance G'.

Nevertheless, our above observation is valid only for a particular graph instance G' at hand. The solution maximizing influence spread on a particular G' does not necessarily maximize influence spread *in expectation*, for any randomly generated graph G'. Yet, it provides a sample of how a diffusion process may look. We propose that, by generating many such graph instances G' and aggregating a score per node from all of them, we can end up with a good approximation of each node's importance in the overall diffusion process. Then, our solution will consist of the k nodes of highest score. This process of assigning scores to nodes is called *score accumulation phase*.

The question that arises is how exactly we should collect scores. The score of a node v should reasonably depend on the number of times v appears among the top-k CCs of a graph instance G'. At the same time, we should take into consideration the fact that, in each graph instance G', *one and only one* node v per top-k connected component CC is sufficient to lead to a maximum-influence solution on G'. Thus, the scores we assign should be *shared* among nodes within the same CC. Putting these two considerations together, we have concluded that a reasonable *score function* for node v is simply $f(v) = 1$; we have experimentally verified that this score function produces better results than other functions we could use, including $f(v) = \frac{1}{|CC_v|}$ and $f(v) = |CC_v|$, where $|CC_v|$ is the size of the CC to which v belongs. Eventually, after R iterations, those nodes that have accumulated the highest scores will be good candidates for inclusion into the seed set S of size k. The produced solution is *non-monotonic*: the solution for size k is not necessarily a subset of the solution of size $k+1$.

While the accumulation phase collects scores that indicate good candidates for inclusion, it suffers from a drawback: neighboring nodes may find themselves in the same CC too often, and collect similar scores as each other, even though only one of them would be in most cases sufficient to bring about the same influence effect that both of them exert. Therefore, we reason that, once a node u is selected into the seed set S, then, for each edge e_{uv} incident on u, the score of node v, adjacent node to u, who is likely to be in the same CC as u, should be penalized in a manner proportional to the score v has accumulated and the probability that e_{uv} is active, $p_{e_{uv}}$. This process of penalizing the scores of selected nodes' neighbors is called *penalization phase*. In detail, in the penalization phase we include nodes to the seed set S by descending score, while, at the same time, for each edge e_{uv} incident on a selected node u, we update the score of node v, adjacent to u, by the formula $s_v = (1 - p_{e_{uv}}) \cdot s_v$.

Algorithm 1: Harvester(G, k)

```
initialize S = ∅;
/* accumulation phase */
s_v = 0 for all v ∈ V;
for i = 1 to R do
    generate G' keeping each edge e∈G with prob. p_e;
    find top-k connected components CC_G' in G';
    for j = 1 to k do
        for node v ∈ CC_G'(j) do
            s_v += 1;
/* penalization phase */
for i = 1 to k do
    select u = argmax(s_v | v ∈ V \ S);
    S = S ∪ {u};
    for each edge e_uv, v ∈ V \ S do
        s_v = (1 − p_{e_uv}) · s_v;
output S;
```

Algorithm 1 presents our Harvester heuristic. Setting $m = |E|$ and $n = |V|$, we compute connected components in an $O(m)$ BFS, and maintain a Fibonacci heap of the top-k components by size. As there are $O(n)$ CCs in any graph instance in the worst case, each iteration of the accumulation phase needs $O(m+n+k\log(n))$ time, hence $O(R(m+n+k\log(n))$ for R iterations. In the penalization phase, we store the score values in a Fibonacci heap, hence need $O(k\log(n)+m)$, where $O(m)$ stands for penalization operations across edges. Thus, the time complexity of Algorithm 1 is $O(R(m+n+k\log(n))$, dominated by the accumulation phase.

IV. Seed Minimization

We now examine the seed minimization problem and propose adaptations of Harvester and other algorithms for it.

A. Problem Definition

As with the influence maximization problem, we are given an undirected graph $G = (V, E)$, where V is a set of nodes and E is a set of edges, and a set of edge probabilities $\{p_{uv}|(u, v) \in E\}$ that determines the influence that nodes u and v have on each other. The influence of nodes on each other follows a diffusion process under the IC model as previously defined. Now we are given a target influence spread $T \leq |V|$, i.e., the expected number of nodes in a graph G that will be activated following a cascade diffusion starting with some initial seed set S_0. Our objective is to minimize the size of a seed set $S_0 \subset V$.

This problem is the dual of the influence maximization problem. In the latter, the number of nodes in S_0 is a constraint

and the achieved influence spread is an objective function, while in the former, the reverse is that case. The dual problem is of no less practical interest than the primal; a company that releases a new product in the market may be interested to achieve adoption from a certain number of customers. In such a case, it is more appealing to provide just enough budget of initial adopters so as to achieve the target spread, than to maximize adoption for a fixed arbitrary number of initial adopters. Next, we present a variation of the Harvester heuristic for seed minimization, as well as the general application of influence maximization algorithms on the dual problem.

B. Algorithms for Seed Minimization

A natural way of solving the seed minimization problem is presented in Algorithm 2. At each iteration, we select the node with the largest marginal influence, i.e., the node that will activate more nodes than any other inactive node in the graph. We keep adding nodes until we reach T.

Algorithm 2: Greedy(G, T)

```
initialize S = ∅;
while σ(S) < T do
    select u = argmax_{v∈V\S} (σ(S ∪ {v}) − σ(S));
    S = S ∪ {u};
output S;
```

Let k be the size of the produced seed set S and assume we need R iterations to compute the expected influence spread for each vertex $v \in V \setminus S$. As it takes $O(m)$ to compute influence spread for one node at one iteration, the running time of Algorithm 2 is $O(knRm)$.

While this greedy algorithm is tailored for seed minimization problem, we can also apply algorithms developed for influence maximization problem in order to get a solution for the dual problem. Let Alg be an algorithm for influence maximization, such that, given a graph G with assigned edge probabilities, and a seed set budget k, Alg returns a subset $S \subseteq V$ of size k. Then, we can apply algorithm Alg to solve the seed minimization problem by *incremental search*, as shown in Algorithm 3. This algorithm is guaranteed to find a solution as long as $T \leq |V|$, while its running time depends on the running time of Alg. Letting $|S^*|$ be the final size of S and U the time complexity of Alg, the time complexity of Algorithm 3 is $O(|S^*|RmU)$.

Algorithm 3: Incremental Search(G, T, Alg)

```
initialize k = 0;
while σ(S) < T do
    k+ = 1;
    S = Alg(G, k);
output S;
```

Still, to reduce the number of influence spread calculations, we can employ binary search, as shown in Algorithm 4. We first find Low and $High$ seed size values for which the spread achieved by Alg are $\sigma(S_{Low}) \leq T \leq \sigma(S_{High})$, and then let the binary search process converge to a seed set size value k for which Alg achieves the target influence spread T. This algorithm will converge to a solution even if the influence spread achieved by Alg is not a monotonic function of seed set size k. However, state-of-the-art influence maximization algorithms do provide such monotonicity, since they build their solutions by incrementally adding more nodes to S in a greedy fashion.

Algorithm 4: Binary Search(G, T, Alg)

```
initialize High = 1;
while σ(S) < T do
    High = 2High;
    S = Alg(G, High);
Low = ⌊High/2⌋;
while Low + 1 < High do
    k = Low + ⌊(High − Low)/2⌋;
    S = Alg(G, k);
    if σ(S) ≥ T then
        High = k;
    else
        Low = k;
output S;
```

C. Reverse Harvester

In contrast to state-of-the-art algorithms for influence maximization, our Harvester algorithm can be conveniently modified so as to be applied to the dual, seed minimization problem. This modification can be brought about as follows: Instead of aiming at the top-k CCs at each iteration for a fixed k, we can find the minimum number t, such that the top-t CCs by size achieve, when put together, a cumulative size at least equal to the target spread T; we emphasize that t is not a fixed parameter of the algorithm, but a variable value computed during each iteration.

Then, if we average the observed values of t over all R iterations, we obtain a good, albeit optimistic, estimate of the seed size value that can achieve target spread T; we say this is an optimistic estimate because, in each of R iterations, the top-t CCs are selected *a posteriori*, after generating a live-edge graph; this state of affairs does not correspond to a real-world scenario. Still, this estimate can be used as an initial value of $High$ in binary search.

Algorithm 5 presents the adaptation of Harvester for seed selection. It consists of three phases. In the *accumulation phase*, we collect scores for nodes in R iterations. At each iteration, as before, we first generate a live-edge graph G' and then assign scores to the nodes in the t top-sized components, which collectively achieve size T (including ties in the t-th position). As before, we store connected components in a priority queue while exploring the graph using DFS or BFS and assign scores to the nodes of the top CCs in this priority queue. We average the values of t over all iterations so as to get a good (optimistic) estimate L of the required initial seed set size.

In the *penalization phase* we select first L nodes by descending score, while penalizing the score of each adjacent neighbor v of a node u across an edge e_{uv} by $p_{e_{uv}}$ of its previous score value s_v, as we did before. Then, we enter the *binary search phase*, in which we search for a seed set size along the list of nodes by descending penalized score,

that achieves just enough influence spread, starting out from size L; at the first stage of this binary search, we increase the seed set size, along with penalization, until its influence spread exceeds T; then we enter a regular binary search operation. As discussed, L is an optimistic estimate, therefore the final selected seed size is usually in the range of $1.5L$ to $1.8L$. Still, as we will show in the next section, the number of required influence spread calculations we need to perform is up to two times less compared to the adaptations of existing heuristics for influence maximization by Algorithm 4.

Algorithm 5: Reverse Harvester(G, T)

initialize $S = \emptyset$;
$s_v = 0$ for all $v \in V$;
$L = 0$;
/* accumulation phase */
for $i = 1$ *to* R **do**
    generate G' by keeping each edge $e \in G$ with prob. p_e;
    select top-t CCs in G such that $\sum_{i=1}^{t} |CC_{G'}(i)| \geq T$;
    for $j = 1$ *to* t **do**
        for *node* $v \in CC_{G'}(j)$ **do**
            $s_v += 1$;
    $L += t/R$
/* penalization phase */
for $i = 1$ *to* L **do**
    select $u = argmax_v\{s_v | v \in V \setminus S\}$;
    $S = S \cup \{u\}$;
    for *each edge* e_{uv}, $v \in V \setminus S$ **do**
        $s_v = (1 - p_{e_{uv}}) \cdot s_v$;
/* binary search phase */
initialize $High = L$;
while $\sigma(S) < T$ **do**
    $High = 2High$;
    while $|S| < High$ **do**
        select $u = argmax_v\{s_v | v \in V \setminus S\}$;
        $S = S \cup \{u\}$;
        for *each edge* e_{uv}, $v \in V \setminus S$ **do**
            $s_v = (1 - p_{e_{uv}}) \cdot s_v$;
$Low = \lfloor \frac{High}{2} \rfloor$;
while $Low + 1 < High$ **do**
    $k = Low + \lfloor \frac{High - Low}{2} \rfloor$;
    $S =$ first k nodes by penalized score s_v;
    if $\sigma(S) \geq T$ **then**
        $High = k$;
    else
        $Low = k$;
output S;

V. Experimental Evaluation

In this section, we present our experimental study comparing our algorithms for influence maximization and seed minimization to the state-of-the-art algorithm in terms of the quality/efficiency tradeoff, IMM [6], on real-world data sets. Our approach is thereby shown to perform significantly better than the current state-of-the-art competing approach on both problems with undirected graphs; on the influence maximization problem, we achieve influence spread that is higher than that achieved by the state of the art methods, while running faster; on the seed minimization problem, our approach requires substantially fewer influence spread evaluations, and, thus takes substantially less time, while also achieving comparable or smaller seed set size for a wide range of target influence spread thresholds.

A. Experiment setup

Data sets. For evaluation purposes we run our algorithms against two real-world undirected, symmetric interaction networks. The first network, Astro, is an academic collaboration network in the Astrophysics section of the e-print arXiv[5]. Our second data set is the Gnutella peer-to-peer file sharing network from August 2002[6]. In this network nodes are hosts in the Gnutella network and edges are connections between the hosts. We refer to these networks as Astro and Gnutella networks. Statistics on them are provided in Table I.

TABLE I – Statistics for two real data sets.

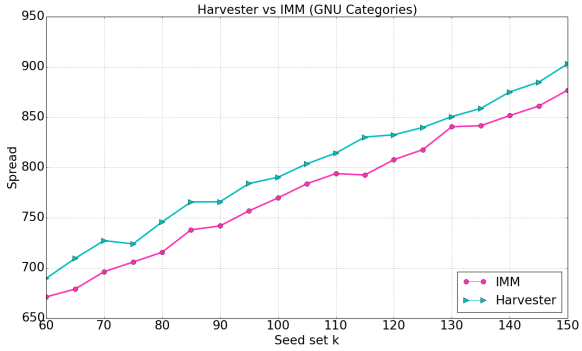
Data set	Astro	Gnutella
Number of nodes	18K	8K
Number of edges	198K	26K
Clustering coefficient	0.63	0.009
Diameter (longest shortest path)	14	10

Cascade models. We compare all algorithms under the general IC model with non-uniform propagation probabilities, using the following three models to assign propagation probabilities to edges of a graph:

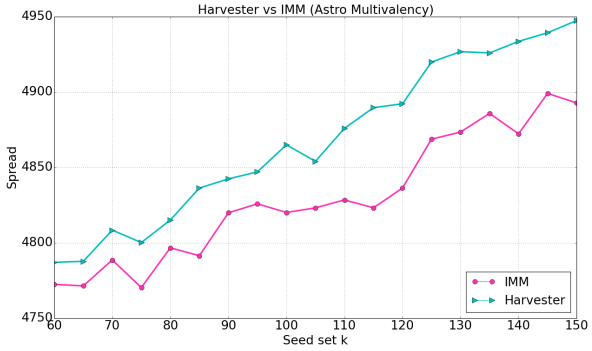
- *MultiValency model*: This is a model similar to the TRIVALENCY model reported in Chen et al. [3]. For every edge of the graph we select a value from the set $\{0.01, 0.02, 0.04, 0.08\}$ uniformly at random. We have tested results for TRIVALENCY model as well, and found similar results.
- *Random model*: Similarly to the MultiValency model, we select a propagation probability uniformly at random from the range $[0, 0.1]$. The average and maximum possible probabilities are larger than those for MultiValency, thus, in general, with this model all tested algorithms result in larger influence spread.
- *Categories model*: Based on the assumption that nodes with high degrees are likely to have larger influence on their neighbors, the Categories model is built in the following manner: let $\{0.01, 0.02, 0.04, 0.08\}$ be a set of possible probability values. We sort nodes by their degrees in ascending order and divide them into four equal-sized chunks (except, maybe, the last chunk). We map the set of values to the chunks so that nodes with low degrees have value 0.01, while nodes with high degree have highest possible value 0.08. For an edge (u, v) we pick a propagation probability at random between value of u and value of v.

We wish to explicitly clarify that we are aware of other methods to assign probabilities to the edges; however, those require additional input to the problem [22]. In addition, we

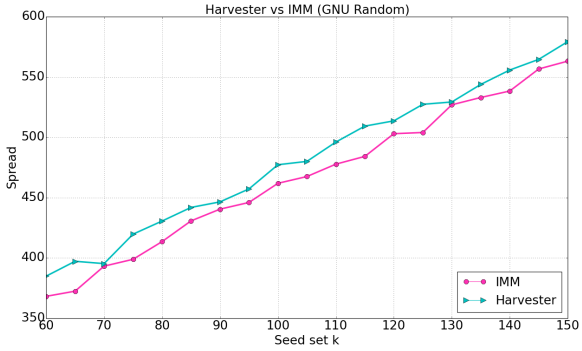
[5] https://snap.stanford.edu/data/ca-AstroPh.html
[6] http://snap.stanford.edu/data/p2p-Gnutella09.html



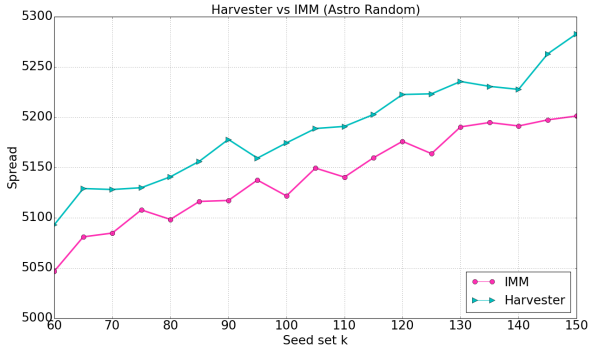
(a) MultiValency, Gnutella network



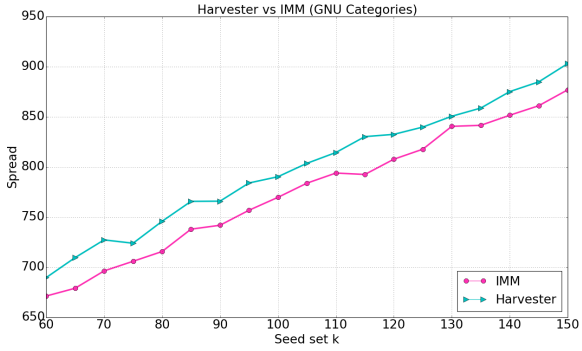
(b) MultiValency, Astro network



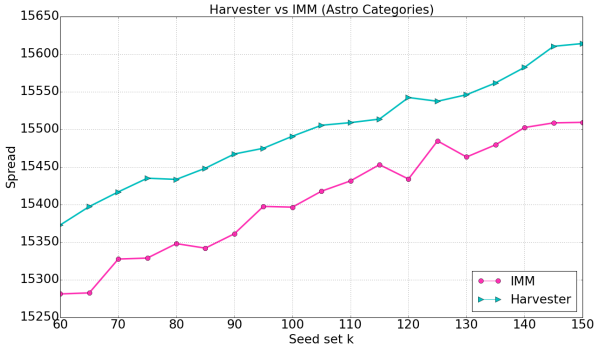
(c) Random, Gnutella network



(d) Random, Astro network



(e) Categories, Gnutella network



(f) Categories, Astro network

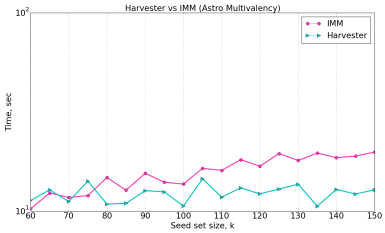
Fig. 1 – Influence spreads in the Gnutella and Astro networks

note that probability values of magnitude higher than 0.1 render the problem less challenging, as a spread of the whole network can be reached with small seed sizes [1].

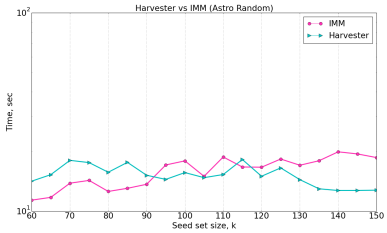
Algorithms. We compare our Harvester heuristic against the current state-of-the-art scalable heuristic for the influence maximization problem; hence the compared algorithms are:

- *Harvester*: Our Algorithm 1 based on the aggregation of live-edge graphs. We use $R = 500$ iterations; we found out that our results showed little difference in influence spread for values beyond that.
- *IMM* IMM, or *Influence Maximization via Martingales* [6], is the established state-of-the-art algorithm for the problem; it has been shown to be superior to TIM, or *Two-phase Influence Maximization* [5], in terms of running time, while achieving *the same* influence spread. In its own turn, TIM has been previously compared to then state-of-the-art algorithms such as *Reverse Influence Sampling* (RIS) [27] (from which it inherits), IRIE [16], and CELF++ [14] (tantamount to the classical Greedy approach), and shown to achieve better performance than those predecessors in terms of *both influence spread and running time.* Therefore, it is reasonable to focus on comparing Harvester's results against those of IMM for evaluation purposes.

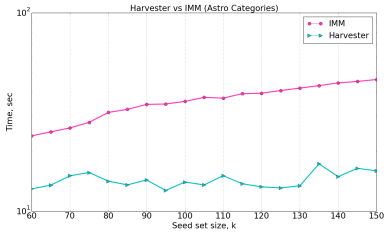
For all algorithms, to calculate the expected influence spread after obtaining their seed set output, we run Monte Carlo simulations with the IC model and non-uniform proba-



MultiValency, Astro network

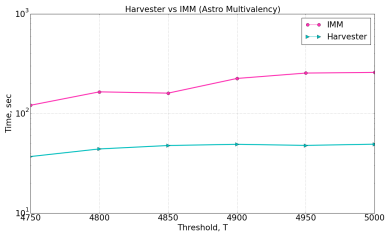


Random, Astro network

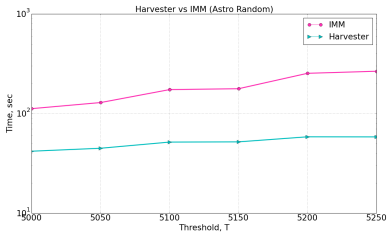


Categories, Astro network

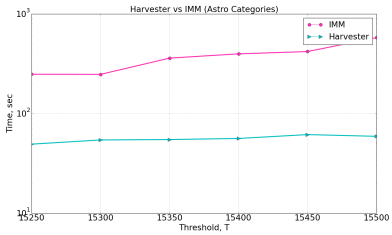
Fig. 2 – Running time for Influence Maximization problem in the Astro network



MultiValency, Astro network



Random, Astro network



Categories, Astro network

Fig. 3 – Running time for Seed Minimization problem in the Astro network

bilities, and take the average of the observed influence spread over 10,000 runs.

B. Experimental Results

We now summarize our experiment results for different data sets under different probability models.

1) Influence Spread: We start out by measuring the influence spread, with the value of seed set size k ranging from 60 to 150. Figure 1 shows our influence spread results with the Gnutella and Astro networks. We observe that Harvester consistently outperforms IMM. For instance, with the Astro network, the average obtained difference in terms of spread is 90, 50, and 40 nodes with the Categories, Random, and Multivalency models, respectively; the obtained picture is similar with the Gnutella network.

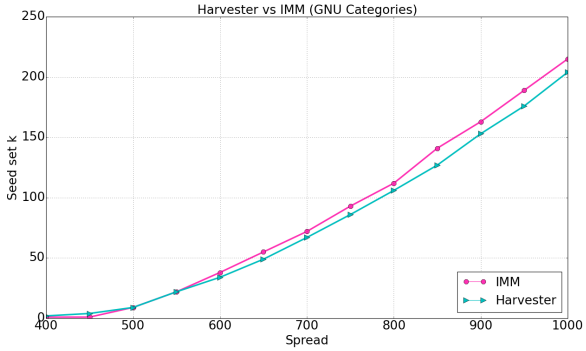
2) Runtime: Next, we measure the runtime that both algorithms need in order to achieve the spread results we have presented. Figure 2 reports our results under the same settings, in which both algorithms need to return seed set S of size k ranging from 60 to 150, on the larger Astro network; the runtime for calculating the final influence spread, which is the same in both cases, is not included. Note that the y-axis is now in logarithmic scale. These experiments were run on an Intel Core i5-2450M CPU machine @ 2.50GHz with 6G memory, while both algorithm were implemented in C++. For all three models, Harvester exhibits lower running time on average than the state-of-the-art algorithm in terms of efficiency; its advantage is magnified with the Categories model, which is more computationally demanding, as more influential hub nodes get higher probability values assigned on their incident edges.

3) Runtime for Seed Minimization Problem: We now study the runtime of the Reverse Harvester variation for the Seed Minization problem. In this case, IMM operates by directly applying binary search on the k parameter that the main IMM algorithm receives as input, until it converges to the desired estimated influence spread value. On the other hand, Reverse Harvester works by going through the phases outlined in Algorithm 5 in Section IV-C, where binary search is only the last step. Figure 3 presents our results. In this case, both algorithms' running time is affected by two factors: (1) the running time of the actual algorithm, and (2) the number of binary search iterations, at the end of which influence spread has to be calculated by a Monte Carlo simulation. The latter is important because Monte Carlo simulations are prohibitively expensive.

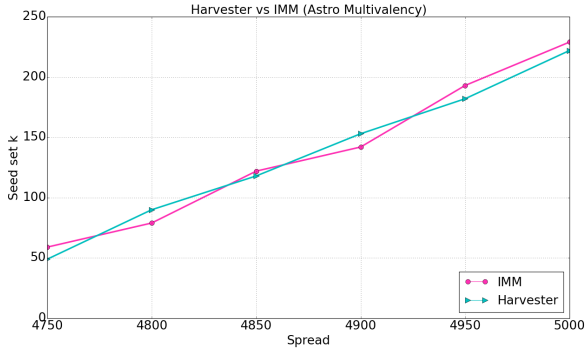
We observe that Harvester's runtime is substantially lower than the one by IMM, as seen on the figure's logarithmic axes, as it performs substantially fewer binary search iterations. In particular, while IMM finishes in hundreds of seconds, Harvester only needs a few seconds to discover a seed set; these results demonstrate that Harvester vastly outperforms the state-of-the-art scalable heuristic on the seed minimization problem, and indicate its robustness and scalability on graphs of large size.

4) Seed set size: Last, we study the actual seed set size achieved by each algorithm when applied on the seed minimization problem, as above. Figure 4 presents our seed set size results on the y-axis, for a given value of target influence spread on the x-axis; for each examined data set and probability model, we consider a range of values of desired influence spread T that presents a challenging problem, in terms of requiring a non-trivial seed set size in order to be achieved.

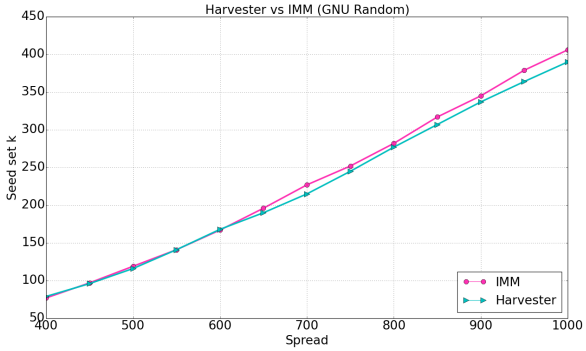
We observe that Reverse Harvester performs better than the adaptation of IMM on the Gnutella network, as it consistently succeeds in finding a seed set of smaller size. In the case



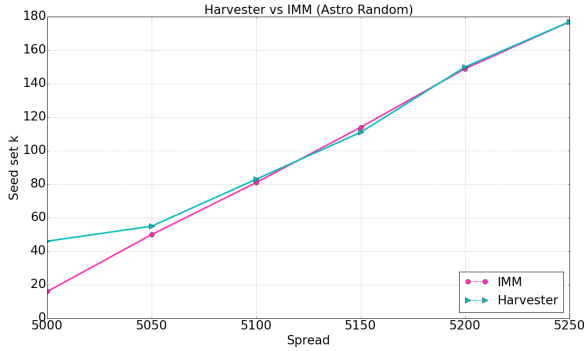
(a) MultiValency, Gnutella network



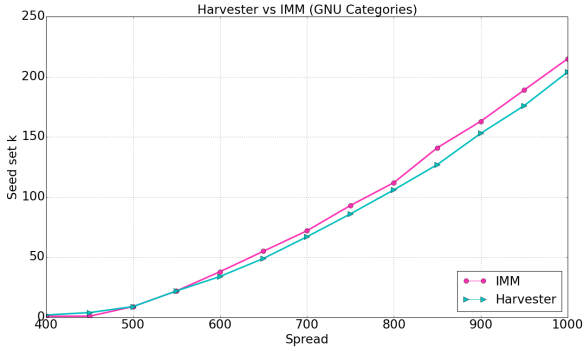
(b) MultiValency, Astro network



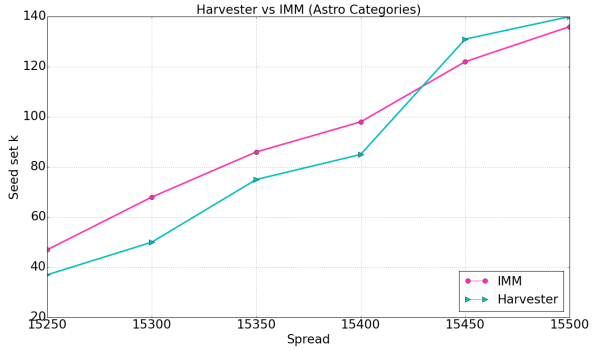
(c) Random, Gnutella network



(d) Random, Astro network



(e) Categories, Gnutella network



(f) Categories, Astro network

Fig. 4 – Seed set size vs. influence spread in the Gnutella and Astro networks

of the Astro network, Reverse Harvester performs in a way comparable to IMM overall, while its performance is more distinguished in the case of the Categories model; the fact that it fails to outperform the solution of IMM in some cases can be attributed to the fact that it aggressively addresses the seed minimization problem in a heuristic manner, while IMM works in round-about fashion, via a solution for the influence maximization problem. Yet Harvester presents itself as an algorithm of choice for the problem due to its overall combination of efficiency and effectiveness.

VI. Conclusions

This paper revisited the influence maximization problem, as well as its dual variant, the seed minimization problem, on the case of symmetric interaction networks, i.e., undirected graphs, a case that has received limited attention to date. We proposed Harvester, an efficient, non-monotonic heuristic, tailored for that case; this heuristic us based on a score aggregation process by multiple live-edge graphs, customized for both problem variants. Our experimental study with real-world data demonstrates that Harvester outperforms the state-of-the-art scalable algorithm, IMM, on undirected networks with both problems, while it is particularly more efficient when applied on seed minimization, thanks to its capacity to inherently provide an estimate of the minimal seed size in advance. Overall, our Harvester algorithms provide efficient, scalable, and robust solutions in comparison to the state-of-the-art scalable approaches with complex graph models. In the future, we plan to investigate how our general approach can be adapted for other influence diffusion models.

REFERENCES

[1] D. Kempe, J. M. Kleinberg, and Éva Tardos, "Maximizing the spread of influence through a social network," in *KDD*, 2003, pp. 137–146.

[2] W. Chen, Y. Wang, and S. Yang, "Efficient influence maximization in social networks," in *KDD*, 2009, pp. 199–208.

[3] W. Chen, C. Wang, and Y. Wang, "Scalable influence maximization for prevalent viral marketing in large-scale social networks," in *KDD*, 2010, pp. 1029–1038.

[4] A. Goyal, W. Lu, and L. V. S. Lakshmanan, "SIMPATH: An efficient algorithm for influence maximization under the linear threshold model," in *ICDM*, 2011, pp. 211–220.

[5] Y. Tang, X. Xiao, and Y. Shi, "Influence maximization: near-optimal time complexity meets practical efficiency," in *SIGMOD*, 2014, pp. 75–86.

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

[7] S. Khanna and B. Lucier, "Influence maximization in undirected networks," in *SODA*, 2014, pp. 1482–1496.

[8] N. Chen, "On the approximability of influence in social networks," *SIAM J. Discrete Math.*, vol. 23, no. 3, pp. 1400–1415, 2009.

[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*, vol. 3, no. 2, pp. 179–192, 2013.

[10] P. Zhang, W. Chen, X. Sun, Y. Wang, and J. Zhang, "Minimizing seed set selection with probabilistic coverage guarantee in a social network," in *KDD*, 2014, pp. 1306–1315.

[11] W. Chen, Y. Yuan, and L. Zhang, "Scalable influence maximization in social networks under the linear threshold model," in *ICDM*, 2010, pp. 88–97.

[12] S. Cheng, H. Shen, J. Huang, G. Zhang, and X. Cheng, "StaticGreedy: solving the scalability-accuracy dilemma in influence maximization," in *CIKM*, 2013, pp. 509–518.

[13] J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. M. VanBriesen, and N. S. Glance, "Cost-effective outbreak detection in networks," in *KDD*, 2007, pp. 420–429.

[14] A. Goyal, W. Lu, and L. V. S. Lakshmanan, "CELF++: Optimizing the greedy algorithm for influence maximization in social networks," in *WWW (Companion Volume)*, 2011, pp. 47–48.

[15] M. Kimura, K. Saito, and R. Nakano, "Extracting influential nodes for information diffusion on a social network," in *AAAI*, 2007, pp. 1371–1376.

[16] K. Jung, W. Heo, and W. Chen, "IRIE: Scalable and robust influence maximization in social networks," in *ICDM*, 2012, pp. 918–923.

[17] J. Kim, S. Kim, and H. Yu, "Scalable and parallelizable processing of influence maximization for large-scale social networks?" in *ICDE*, 2013, pp. 266–277.

[18] S. Cheng, H. Shen, J. Huang, W. Chen, and X. Cheng, "IMRank: influence maximization via finding self-consistent ranking," in *SIGIR*, 2014, pp. 475–484.

[19] Y. Wang, G. Cong, G. Song, and K. Xie, "Community-based greedy algorithm for mining top-k influential nodes in mobile social networks," in *KDD*, 2010, pp. 1039–1048.

[20] Q. Jiang, G. Song, G. Cong, Y. Wang, W. Si, and K. Xie, "Simulated annealing based influence maximization in social networks," in *AAAI*, 2011.

[21] W. Chen, A. Collins, R. Cummings, T. Ke, Z. Liu, D. Rincón, X. Sun, Y. Wang, W. Wei, and Y. Yuan, "Influence maximization in social networks when negative opinions may emerge and propagate," in *SDM*, 2011, pp. 379–390.

[22] A. Goyal, F. Bonchi, and L. V. S. Lakshmanan, "A data-based approach to social influence maximization," *PVLDB*, vol. 5, no. 1, pp. 73–84, 2011.

[23] H. Li, S. S. Bhowmick, and A. Sun, "CINEMA: conformity-aware greedy algorithm for influence maximization in online social networks," in *EDBT*, 2013, pp. 323–334.

[24] O. Ben-Zwi, D. Hermelin, D. Lokshtanov, and I. Newman, "An exact almost optimal algorithm for target set selection in social networks," in *EC*, 2009, pp. 355–362.

[25] S. Goldberg and Z. Liu, "The diffusion of networking technologies," in *SODA*, 2013, pp. 1577–1594.

[26] S. Lei, S. Maniu, L. Mo, R. Cheng, and P. Senellart, "Online influence maximization," in *KDD*, 2015, pp. 645–654.

[27] C. Borgs, M. Brautbar, J. T. Chayes, and B. Lucier, "Maximizing social influence in nearly optimal time," in *SODA*, 2014, pp. 946–957.