Distance-Based Influence in Networks: Computation and Maximization

Edith Cohen
Tel Aviv University
Google Research
edith@cohenwang.com

Daniel Delling
Sunnyvale
USA
daniel.delling@gmail.com

Thomas Pajor
Cupertino
USA
thomas@tpajor.com

Renato F. Werneck
San Francisco
USA
rwerneck@acm.org

ABSTRACT

A premise at a heart of network analysis is that entities in a network derive utilities from their connections. The *influence* of a seed set S of nodes is defined as the sum over nodes j of the *utility* of S to j. *Distance-based* utility, which is a decreasing function of the distance from S to j, was explored in several successful research threads from social network analysis and economics: Network formation games [Bloch and Jackson 2007], Reachability-based influence [Richardson and Domingos 2002; Kempe et al. 2003]; "threshold" influence [Gomez-Rodriguez et al. 2011]; and *closeness centrality* [Bavelas 1948].

We formulate a model that unifies and extends this previous work and address the two fundamental computational problems in this domain: *Influence oracles* and *influence maximization* (IM). An oracle performs some preprocessing, after which influence queries for arbitrary seed sets can be efficiently computed. With IM, we seek a set of nodes of a given size with maximum influence. Since the IM problem is computationally hard, we instead seek a *greedy sequence* of nodes, with each prefix having influence that is at least $1-1/e$ of that of the optimal seed set of the same size. We present the first highly scalable algorithms for both problems, providing statistical guarantees on approximation quality and near-linear worst-case bounds on the computation. We perform an experimental evaluation which demonstrates the effectiveness of our designs on networks with hundreds of millions of edges.

1. INTRODUCTION

Structural notions of the *influence* of a set of entities in a network which are based on the *utility* an entity derives from its connectivity to others, are central to network analysis and were studied in the context of social and economic models [2,3,13,20,23,25,29,32] with applications that include ranking, covering, clustering, active semi-supervised learning, and studying diffusion and network formation games. Formally, influence can be expressed in terms of utilities u_{ij} between ordered pairs of entities. The influence of a single entity i, also known as its *centrality,* is the sum $\mathsf{Inf}(i) = \sum_j u_{ij}$ over entities j, of the utility of i to j. The influence of a set S of entities is the sum over entities of the highest utility match from S:

$$\mathsf{Inf}(S) = \sum_j \max_{i \in S} u_{ij} \ .$$

One of the simplest and more popular definitions of influence relies on *reachability-based* utility [21, 25, 30]. The network here is a directed graph, where entities correspond to nodes. We have $u_{ij} = 1$ when the node j is reachable from node i. Therefore, the influence of S is the number of entities reachable from S. A powerful enhancement of this model is to allow for multiple *instances*, where each instance is a set of directed edges, or for a distribution over instances, and accordingly, define u_{ij} as the respective average or expectation. The popular Independent Cascade (IC) model of Kempe et al. [25] uses a distribution defined by a graph with independent inclusion probabilities for edges.

More expressive utility is based on shortest-path distances [2, 3, 13, 20, 25, 29, 32]. Specifically, the utility is $u_{ij} = \alpha(d_{ij})$, where $\alpha(d) \geq 0$ is a non-increasing function [3, 13, 16, 31] applied to the respective shortest-path distance. Reachability-based utility falls out as a special case, using $\alpha(x) = 1$ for finite x and $\alpha(+\infty) = 0$. Popular kernels include exponential decay $\alpha(x) = \exp(-\lambda x)$, polynomial decay $\alpha(x) = 1/\mathrm{poly}(x)$, Gaussian $\alpha(x) = \exp(-\lambda x^2)$, and threshold, which is obtained using $\alpha(x) = 1$ when $x \leq T$ and $\alpha(x) = 0$ otherwise. This variety, used in practice, demonstrates the value of this modeling flexibility.

A distance-based model for information diffusion in social networks was recently proposed by Gomez-Rodriguez et al. [22]. In their model, the seed set S corresponds to the "infected" nodes and edge lengths correspond to propagation time. The shortest paths distance from S to a node v corresponds to the elapsed time until v is "infected." The decay function models the amount by which slower propagating is less valuable. Again, the model effectiveness is enhanced by working with multiple instances (each instance is a set of directed edges with lengths), or with a distribution over such instances, and defining the utility as the average of $\alpha(d_{ij})$ over instances or as the expectation $u_{ij} = \mathsf{E}[\alpha(d_{ij})]$. A simple but powerful model associates independent randomized edge lengths (REL) with live edges [1, 11, 18, 22], using exponential [1, 11, 22] or Weibull [18] distributions. Randomization is so effective with reachability or distance-based utilities because the quality of connectivity depends not only on the existence of a path or the shortest-path length, but also is higher when there are multiple independent paths [11], and with randomization, utilities are higher with multiple paths. Moreover, introduction of random noise is a standard method to prevent overfitting.

The models of Gomez-Rodriguez et al. and subsequent work on distance-based diffusion [18, 22] was focused specifically on threshold decay functions, where for a threshold parameter T, $u_{ij} = 1$ only when the distance is at most T. Distance-based utility with smooth decay functions, which we study here, is motivated by physical laws, and extensively studied in data analysis [31] and networks [3, 13, 23, 29]. In particular, distance-based influence generalizes the distance-decaying variant [5, 9, 13, 17, 29] of closeness centrality [2], which was studied with exponential, harmonic, threshold, and general decay functions.

The two fundamental algorithmic problems in applications of influence are *influence computation* and *influence maximization* (IM).

Influence computation is the problem of computing the influence

of a specified seed set S of nodes. This can be done using multiple single-source shortest-paths computations from the seed set, but the computation does not scale well when there are many queries on very large networks. Cohen et al. for reachability-based influence [12] and Du et al. for threshold influence [18] designed *influence oracles* which preprocess the input so that influence queries $\mathsf{Inf}(\mathscr{G},S)$ for a specified seed set S can be approximated quickly. In both cases, a MinHash sketch, based on [8], is computed for each node so that the influence of a seed set S can be estimated from the sketches of the nodes in S.

Influence maximization is the problem of finding a seed set $S \subset V$ with maximum influence, where $|S| = s$ is given. Since reachability-based and threshold influence [22, 25] are special cases, we know that distance-based influence maximization with general α is NP-complete and hard to approximate to anything better than $1-(1-1/s)^s$ of the optimum for a seed set of size s (the hardness result is asymptotic in s) [19]. Fortunately, from monotonicity and submodularity of these influence functions [22, 25], we obtain that the greedy algorithm (GREEDY), which iteratively adds to the seed set the node with maximum marginal influence, is guaranteed to provide a solution that is at least $1-(1-1/s)^s > 1-1/e$ of the optimum [28]. This (worst-case) guarantee holds for every prefix size of the sequence of seeds reported, which means that the GREEDY sequence approximates the Pareto front of the trade-off of seed set size versus its influence. The Pareto front is important on its own as a characterization of the influence coverage of the network and to facilitate finding bi-criteria sweet spots between the size of the seed set and its coverage.

In terms of solution quality, also in practice, GREEDY had been the gold standard for submodular maximization. Ideally, we seek near-linear computation with greedy-like approximation guarantees. But exact greedy does not scale even with various optimizations [27], even for reachability-based influence on a single instance. This prompted the development of scalable heuristics [24] and of a notable approach [6, 35] of running the greedy algorithm on (pre) sampled or sketched [7, 8] influence sets. Inherently, however, pre-computed near-linear size samples can not provide greedy-like quality guarantees beyond a small number of seeds, even without randomization. The only algorithm that is both near-linearly with input size and provides greedy-like quality guarantees on a full sequence of seeds is SKIM, which works by maintaining samples with respect to a *residual problem* that are maintained as seeds are selected [12].[1] For threshold influence, existing maximization algorithms are by Du et al. [18] based on sketches [8] and by [34] which extends the pre-sampling approach for reachability-based influence [35]. Again, such designs, when limited to a near-linear computation, can not provide greedy-like approximation guarantees beyond a small number of seeds, even when edge lengths are not randomized.

Contributions. This paper is an overview of our contributions. We refer the reader to a full version in http://arxiv.org/abs/1410.6976. The paper is organized as follows. Our distance-based influence model is presented in Section 2. We define exact and approximate GREEDY sequences and establish approximation guarantees of both. We then formulate a representation of a *residual problem* with respect to a seed set S, which facilitates the greedy computation of the next seed. Finally, we establish a probabilistic bound on residual updates, which magically holds for the approximate but not for the exact greedy. This bound is a critical component in obtaining near-linear computation of a full approximate greedy sequence.

Our results for the threshold influence model [18,22] are overviewed in Section 3. We extend the (approximate) influence oracles and the SKIM reachability-based influence maximization algorithm [12], to obtain oracles and T-SKIM for threshold influence. Algorithmically, the extension replaces the reachability searches in [12] with Dijkstra computations that are pruned at the threshold distance. The statistical guarantees on approximation quality of the oracles and of the approximate greedy sequence are inherited from SKIM. Our worst-case running time analysis, however, which establishes that T-SKIM computes the full approximate greedy sequence in near-linear time, required our novel probabilistic bound on residual updates. Our results for threshold influence significantly improve over previous results, both in terms of theoretical bounds and in practice and also serve as a warmup before treating the general influence model.

Our *distance-based influence oracles* are presented in Section 4. This is a practically important contribution which again improves significantly over previous work [18]. The oracle take as input a seed set S and (*any*) decay function α, which can be specified at query time and returns the estimated influence. As explained earlier, many different decay/kernel functions are used extensively in practice, which makes the flexibility of the oracle to handle arbitrary α valuable. Our oracle computes a novel sketch for each node; the *combined All-Distances sketch* (cADS), which generalizes All-Distances Sketches (ADS) [4, 8, 9, 13, 14], used for closeness centrality computation, to multiple instances or probabilistic models. These per-node sketches have expected size at most $k\ln(n\min\{k,\ell\})$ (with good concentration), where n is the number of nodes, ℓ is the number of instances, and k is a sketch parameter that determines a trade-off between information and the amounts of computation and space required. We estimate the distance-based influence of a seed set S from the sketches of the nodes in S. The estimates, also using simpler estimators, have worst-case coefficient of variation (CV) $\leq 1/\sqrt{k-2}$, but we derive estimators that optimally use the information in the sketch and can be much tighter for seed sets with diverse coverage. In particular, we use HIP probabilities [9] with the L* estimator [10] and also have a worst-case CV bound of $\leq 1/\sqrt{2k-1}$.

Our technically most challenging contribution is α-SKIM, which is the first scalable influence maximization algorithm that applies with general decay functions α. We refer the reader to the full version for details. Our design is a strong contribution from both theoretical and practical perspectives, providing a novel near-linear worst-case bound on the computation, performance guarantees that nearly match those of exact GREEDY, and a scalable implementation that runs on large networks. The heart of our design is a novel algorithmic technique of efficiently maintaining weighted samples from influence sets that allows us to accurately estimate marginal influences as nodes are added to the seed set.

Section 5 presents a comprehensive experimental evaluation of our algorithms. For threshold influence oracles and maximization, we obtain three orders of magnitude speedups over the algorithms of Du et al. [18], with no loss in quality. Even though both approaches apply the sketches of Cohen [8], we are able to obtain improvements by working with combined sketches, applying better estimators, and, in our IM algorithm, only computing sketches to the point needed to determine the next node. We also show that the generalization to arbitrary decay functions α is only slightly slower, and can easily handle graphs with hundreds of millions of edges.

2. DISTANCE-BASED MODEL

For a set of entities V, *utility* u_{ij} for $i, j \in V$, and a *seed set* $S \subset V$,

[1] SKIM guarantees hold for inputs specified as a set of instances. It is not known if near-linear time algorithms with greedy-like guarantees exist for the IC model.

we define the *influence* of S as

$$\mathsf{Inf}(S) = \sum_{j \in V} \max_{i \in S} u_{ij} \ .$$

Distance-based utility is defined with respect to a non-increasing function α such that $\alpha(\infty) \equiv 0$ and (a set or distribution over) edge-weighted graphs $G = (V, E, w)$, where the nodes correspond to entities and edges $e \in E$ have lengths $w(e) > 0$.

We refer to a single graph G as an *instance*. We denote by d_{ij} the shortest-path distance in G from i to j. When there is no path from i to j in G we define $d_{ij} \equiv \infty$. For a set of nodes S, we let $d_{Sj} = \min_{i \in S} d_{ij}$ be the shortest-path distance in G from S to j.

The utility of node i to node j with respect to a single instance is defined as $u_{ij} = \alpha(d_{ij})$, yielding the influence function

$$\mathsf{Inf}(G, S) = \sum_{j \in V} \alpha(d_{Sj}).$$

For a set $\mathscr{G} = \{G^{(t)}\}$ of $\ell \geq 1$ instances $G^{(t)} = (V, E^{(t)}, w^{(t)})$, we define the utility of i to j as the average $u_{ij} = \frac{1}{\ell} \sum_{t=1}^{\ell} \alpha(d_{ij}^{(t)})$ and accordingly the influence is the average:

$$\mathsf{Inf}(\mathscr{G}, S) = \mathsf{Inf}(\{G^{(t)}\}, S) = \frac{1}{\ell} \sum_{t \in [\ell]} \mathsf{Inf}(G^{(t)}, S). \tag{1}$$

Our algorithms work with inputs specified as a set of one or more instances and with respect to the influence function (1).

A set of instances can be derived from traces or generated by Monte-Carlo simulations of a probabilistic model $\mathscr{G}$. Such a model defines a distribution over instances $G \sim \mathscr{G}$ which share a set V of nodes. The respective utility is then $u_{ij} = \mathsf{E}[\alpha(d_{ij})]$ and the influence of $\mathscr{G}$ is then the expectation

$$\mathsf{Inf}(\mathscr{G}, S) = \mathsf{E}_{G \sim \mathscr{G}} \mathsf{Inf}(G, S). \tag{2}$$

When Inf is well concentrated around its expectation, a small number of simulations suffices to approximate the results.

2.1 The Exact Greedy Sequence

We present the exact greedy algorithm for the distance-based influence objective $\mathsf{Inf}(\{G^{(t)}\}, S)$, as defined in Equation (1). GREEDY starts with an empty seed set $S = \emptyset$. In each iteration, it adds the node with maximum marginal gain, that is, the node i that maximizes the influence of $S \cup \{i\}$. GREEDY thus produces a sequence of nodes, providing an approximation guarantee for the seed set defined by each prefix.

We now elaborate on the computation of the marginal gain of i given S. To do so efficiently as S grows, we work with a *residual problem*. We denote the residual problem of $\mathscr{G}$ with respect to seed set S as $\mathscr{G}|S$. The influence of a set of nodes U in the residual problem is equal to the marginal influence in the original problem:

$$\mathsf{Inf}(\mathscr{G}|S, U) = \mathsf{Inf}(\mathscr{G}, S \cup U) - \mathsf{Inf}(\mathscr{G}, S).$$

A residual problem has a slightly more general specification. The input has the form $(\mathscr{G}, \delta)$, where $\delta_j^{(t)} \geq 0$ maps node-instance pairs (j,t) to nonnegative numbers. The δ values we use for the residual problem $\mathscr{G}|S$ are the respective distances from the seed set (but can be truncated without violating correctness at any distance x in which $\alpha(x) = 0$):

$$\delta_j^{(t)} = d_{Sj}^{(t)} \equiv \min_{i \in S} d_{ij}^{(t)}.$$

When the seed set is empty or the node j is either not reachable from S in instance t or has distance $d_{Sj}^{(t)} > \sup_x \{\alpha(x) > 0\}$, we can use $\delta_j^{(t)} = \infty$ or any $\delta_j^{(t)} > \sup_x \{\alpha(x) > 0\}$, which is equivalent.

We now extend the influence definition for inputs of the form $(\mathscr{G}, \delta)$. For a node i and a node-instance pair (j,t), the contribution of instance t to the utility of i to j which is also the contribution of (j,t) to the influence of i is:

$$\Delta_{ij}^{(t)} \equiv \max\{0, \alpha(d_{ij}^{(t)}) - \alpha(\delta_j^{(t)})\}. \tag{3}$$

The influence of i in the residual problem is the (normalized) sum of these contributions over all nodes in all instances:

$$\mathsf{Inf}((\mathscr{G}, \delta), i) \equiv \frac{1}{\ell} \sum_t \sum_j \Delta_{ij}^{(t)}. \tag{4}$$

It is not hard to verify the following.

LEMMA 2.1. *For any set of nodes U, the influence of U in $\mathscr{G}|S$ is the same as marginal influence of U with respect to S in $\mathscr{G}$.*

Given a residual input $(\mathscr{G}, \delta)$, the influence of a node i can be computed using a pruned application of Dijkstra's algorithm from i. The pruning is performed for efficiency reasons by avoiding expanding the search in futile directions. In particular, we can always prune at distance d when $\alpha(d) = 0$ or when $d \geq \delta_j^{(t)}$. The correctness of the pruning follows by observing that all nodes i Dijkstra could reach from the pruned node have $\Delta_{ij}^{(t)} = 0$.

At each step, GREEDY selects a node with maximum influence in the residual input. It then updates the distances δ so that they capture the residual problem $\mathscr{G}|S \cup \{i\}$. For details see the full version.

2.2 Approximate Greedy Sequences

APPROXIMATE GREEDY is similar to exact GREEDY, but in each iteration, instead of selecting a seed node with maximum marginal gain, we select a seed node with marginal contribution that is within a small relative error ε of the maximum with high probability. It also suffices to require that the relative error is bounded by ε in expectation and is concentrated, that is, $\forall a > 1$, the probability of error exceeding $a\varepsilon$ decreases exponentially in a. It turns out that the approximation ratio of APPROXIMATE GREEDY is $1 - (1 - 1/s)^s - O(\varepsilon)$ with corresponding guarantees [12].

The SKIM algorithm applies approximate greedy to reachability-based influence. It works with partial sketches (samples) from "influence sets" of nodes to determine a node with approximately maximum marginal gain. These partial sketches need to be updated very efficiently after each seed is selected. A critical component for the scalability and accuracy of SKIM is sampling with respect to the residual problem. This is because the marginal influence of a node can be much smaller than its initial influence and we can get good accuracy with a small sample only if we use the residual problem.

The residual problem (and current samples) are updated after each seed selection, both with exact and approximate GREEDY. We now consider the total number of edge traversals used in these updates. With reachability-based influence, with exact or approximate GREEDY, the number of traversals is linear in input size: This is because there can be at most one search which progresses through a node in each instance. Once a node is reachable from the seed set in that instance, it is influenced, and everything reachable from the node in the same instance is reachable, and influenced, as well. So these nodes never need to be visited again and can be removed.

This is not true, however, for distance-based influence: For each node-instance pair (j,t), the distance $\delta_j^{(t)}$ can be updated many times. Moreover, when the distance to the seed set decreases, as a result of adding a new seed node, the node j and its outgoing edges in the instance t are traversed. Therefore, to bound the computation,

and in particular, the number of edge traversals performed, we must bound the number of updates of $\delta_j^{(t)}$.

For exact GREEDY, there are pathological inputs with $\Omega(sn)$ updates, which roughly translates to $\Omega(s|E|)$ edge traversals, even on a single instance and with threshold influence. Remarkably, we show that *even in the worst-case*, our APPROXIMATE GREEDY selection guarantees a near-linear number of updates (proof provided in the full version):

THEOREM 2.1. *Suppose that the approximate greedy selection has the property that for some ε, the next seed is selected in a near uniform way from all nodes with marginal influence that is at least $(1-\varepsilon)$ of the maximum. Then the expected total number of updates of $\delta_j^{(t)}$ at a node-instance pair (j,t) is bounded by $O(\varepsilon^{-1}\log^2 n)$.*

3. THRESHOLD INFLUENCE

We present both an oracle and an approximate greedy IM algorithm for threshold influence [18, 22]. In this model, a node i has utility $u_{ij} = 1$ to node j if j is within distance at most T from i and $u_{ij} = 0$ otherwise. In terms of our general model, we have $\alpha(x) = 1$ when $x \le T$ and $\alpha(x) = 0$ otherwise. Threshold influence resembles reachability-based influence in that utilities and marginal utilities are always $\{0,1\}$. This property greatly simplifies the design of the algorithms.

3.1 Threshold-Influence Oracle

Our influence oracle for a prespecified threshold T generalizes the reachability-based influence oracle of Cohen et al. [12]. The reachability-based influence oracle preprocesses the input to compute a *combined reachability sketch* for each node. Each node-instance pair is assigned a random permutation rank (a number in $[n\ell]$) and the combined reachability sketch of a node i is a set consisting of the k smallest ranks amongst node-instance pairs (j,t) such that j is reachable from i in instance t. This is also called a bottom-k sketch of reachable pairs. The oracle uses the sketches of the nodes in S to estimate their influence by applying the union size estimator for bottom-k sketches [15]. The combined reachability sketches are built by first computing a set of reachability sketches (one for each node) [8] in each instance and then combining, for each node, the sketches obtained in different instances to obtain one size-k sketch. In turn, the computation for each instance uses reverse (backward) reachability computations. The algorithm of Cohen [8] initiates these reversed reachability searches from all nodes in a random permutation order. These searches are pruned at nodes already visited k times.

For threshold influence, we instead consider a pair (j,t) reachable from i if $d_{ij}^{(t)} \le T$. We then compute for each node the bottom-k sketch of these "reachable" pairs under the modified definition. The oracle estimator [15] is the same one used for the reachability-based case; the estimate has (worst-case) CV that is at most $1/\sqrt{k-2}$ with good concentration. The computation of the sketches is nearly as efficient as for the reachability-based case. Instead of using reverse reachability searches, for threshold influence we use *reverse Dijkstra* computations (single-source shortest-path searches on the graph with reversed edges). These computations are pruned both at distance T and (as with reachability sketches) at nodes already visited k times. The sets of sketches obtained for the different instances are combined as in [12] to obtain a set of combined sketches (one combined sketch with k entries for each node).

The running time is dominated by the computation of the sketches. The preprocessing computation is $O(k\sum_{t=1}^{\ell}|E^{(t)}|\log n)$, the sketch representation is $O(kn)$, and each influence query for a set S takes $O(|S|k\log|S|)$ time.

As in [12] for reachability-based influence in the IC model, we can construct an oracle of the same size and approximate guaranees for a distance-based IC model. The algorithm, however, uses kn reverse searches (see full version for details).

Algorithm 1: Threshold IM (T-SKIM)

Input: Directed graphs $\{G^{(t)}\}$, threshold T, parameter k
Output: Sequence of node and marginal influence pairs

```
// Initialization
forall the node/instance pairs (u,t) do δ[u,t] ← ∞
forall the nodes v do size[v] ← 0
index ← hash map of node-instance pairs to nodes
seedlist ← ⊥    // List of seeds & marg. influences
rank ← 0
shuffle the nℓ node-instance pairs (u,t)
// Compute seed nodes
while |seedlist| < n do
    while rank < nℓ do                    // Build sketches
        rank ← rank + 1
        (u,t) ← rank-th pair in shuffled sequence
        if δ[u,t] < ∞ then skip  // Pair (u,t) is covered
        run Dijkstra from u in reverse graph G^(t), during which
        foreach scanned node v in distance d do
            if d > T then prune      // Prune at depth T
            size[v] ← size[v] + 1
            index[u,t] ← index[u,t] ∪ {v}
            if size[v] = k then
                x ← v                        // Next seed node
                abort sketch building
    if all nodes u have size[u] < k then
        x ← argmax_{u∈V} size[u]
    I_x ← 0                              // The coverage of x
    forall the instances t do            // Residual problem
        run Dijkstra from x in forward graph G^(t), during which
        foreach scanned node v in distance d do
            if δ[v,t] ≤ d or d > T then prune
            if δ[v,t] = ∞ then I_x ← I_x + 1
            δ[v,t] ← d
            forall the nodes w in index[v,t] do
                size[w] ← size[w] − 1
            index[v,t] ← ⊥       // Erase (v,t) from index
    seedlist.append(x, I_x/ℓ)
return seedlist
```

3.2 Threshold-Influence Maximization

Our algorithm for threshold influence maximization, which we call T-SKIM, generalizes SKIM [12], which was designed for reachability-based influence. A pseudocode for T-SKIM is provided as Algorithm 1.

Our algorithm T-SKIM builds sketches, but only to the point of determining the node with maximum estimated influence. We then

compute a residual problem which updates the sketches. T-SKIM build sketches using reverse single-source shortest path computations that are pruned at distance T (depth-T Dijkstra). As with exact greedy for distance-based influence (Section 2), T-SKIM maintains a residual problem. This requires updating the distances $\delta[j,t] = d_{Sj}^{(t)}$ from the current seed set S, as in `AddSeed`(i), and also updating the sketches to remove the contributions of pairs that are already covered by the seed set.

The (worst-case) estimation quality guarantee of T-SKIM is similar to that of SKIM. When using $k = O(\varepsilon^{-2}\log n)$ we obtain that, with high probability (greater than $1-1/\text{poly}(n)$), for all $s \geq 1$, the influence of the first s selected nodes is at least $1-(1-1/s)^s-\varepsilon$ of the maximum influence of a seed set of size s. The computation time is near-linear and analysis is provided in the full version.

4. INFLUENCE ORACLE

We now present our oracle for distance-based influence, as defined in Equation (1). We preprocess the input $\mathscr{G}$ to compute a sketch X_v for each node v. Influence queries, which are specified by a seed set S of nodes and *any* function α, can be approximated from the sketches of the query seed nodes.

We present here the definition of the sketches. For an input specified as either a set of instances or as a distance-based IC model, each sketch X_v has a (well concentrated) expected size that is at most $k\ln(nk)$ (and $k\ln(n\min\{k,\ell\})$ when using ℓ instances). The total storage of our oracle is therefore $O(nk\log(nk))$ (and $nk\ln(n\min\{k,\ell\})$ when using ℓ instances).

In the full version we detail our influence estimator, which optimally uses the information in the sketch, and the algorithm to compute the sketches (preprocessing). We show that for a set of ℓ instances $\mathscr{G} = \{G^{(t)}\}$, the expected time is $O(k\sum_{t=1}^{\ell}|E^{(t)}|\log n)$. We establish the following worst-case bounds on estimate quality:

THEOREM 4.1. *Influence queries* $\mathsf{Inf}(\mathscr{G},S)$*, specified by a set* S *of seed nodes and a function* α*, can be estimated in* $O(|S|k\log n)$ *time from the sketches* $\{X_u \mid u \in S\}$*. The estimate is nonnegative and unbiased, has CV* $\leq 1/\sqrt{2k-2}$*, and is well concentrated (the probability that the relative error exceeds* $a/\sqrt{k}$ *decreases exponentially with* $a > 1$*).*

Our *combined* All-Distances Sketches (cADS) are a multi-instance generalization of All-Distances Sketches (ADS) [8, 9, 14] and build on the related combined reachability sketches [12] used for reachability-based influence.

The cADS sketches are randomized structures defined with respect to random rank values $r_u^{(t)} \sim U[0,1]$ associated with each node-instance pair (u,t). To improve estimation quality in practice, we restrict ourselves to a particular form of *structured permutation ranks* [12]: For a set of instances, the ranks are a permutation of $1,\ldots,n\min\{\ell,k\}$, where each block of positions of the form $in,(i+1)n-1$ (for integral i) corresponds to an independent random permutation of the nodes. For each node u, the instances i_j in $r_u^{(i_j)}$, when ordered by increasing rank, are a uniform random selection (without replacement).

For each node v, cADS(v) is a set of rank-distance pairs of the form $(r_u^{(t)}, d_{vu}^{(t)})$ which includes $\min\{\ell,k\}$ pairs of distance 0, that is, all such pairs if $\ell \leq k$ and the k smallest rank values otherwise. It also includes pairs with positive distance when the rank value is at most the kth smallest amongst closer nodes (across all instances). Formally,

$$\text{cADS}(v) = \begin{cases} \{(r_v^{(t)},0) \mid r_v^{(t)} \in \text{BOTTOM-}k\{r_v^{(j)} | j \in [\ell]\}\} \\ \{(r_u^{(t)}, d_{vu}^{(t)}) \mid r_u^{(t)} < \mathrm{k}^{\text{th}}_{(y,j)|d_{vy}^{(j)}<d_{vu}^{(t)}} r_y^{(j)}\} \end{cases} . \quad (5)$$

Here BOTTOM-k refers to the smallest k elements in the set and k^{th} denotes the kth smallest element in the set. When there are fewer than k elements, we define k^{th} as the domain maximum. For the purpose of sketch definition, we treat all positive distances across instances as unique.

Note that we can also define cADS sketches with respect to a probabilistic model. The definition emulates working with an infinite set of instances generated according to the model. Since there are at most nk distinct rank values in the sketches, and they are all from the first nk structured permutation ranks, the entries in the sketches are integers in $[nk]$.

5. EXPERIMENTS

Our algorithms were implemented in C++ and compiled using Visual Studio 2013 with full optimization. Our test machine runs Windows 2008R2 Server and has two Intel Xeon E5-2690 CPUs and 384 GiB of DDR3-1066 RAM. Each CPU has 8 cores (2.90 GHz, 8×64 kiB L1, 8×256 kiB, and 20 MiB L3 cache). For consistency, all runs are sequential.

The datasets in our experiments are obtained from the SNAP [33] project and represent *social* (Epinions, Slashdot , Gowalla , TwitterFollowers , LiveJournal , Orkut) and *collaboration* (AstroPh) networks. All these graphs are unweighted.

Unless otherwise mentioned, we test our algorithms using $\ell = 64$ independent instances generated from the graph by assigning independent random length to every edge according to an exponential distribution with expected value 1 [1, 11, 22]. We use ADS parameter $k = 64$.

5.1 Distance-Based Influence Maximization

We start with the Influence Maximization problem. Recall that we consider two variants of this problem: threshold influence and general distance-based influence. We discuss each in turn.

5.1.1 Threshold Influence

Our first experiment considers the performance of T-SKIM (Section 3), which finds a sequence of seed nodes such that each prefix of the sequence approximately maximizes the influence. Our results are summarized in Table 3. For each dataset, we first report its total numbers of nodes and edges. This is followed by the total influence (as a percentage of the total number of nodes of the graph) of the seed set found by our algorithm. We report figures for 50 and 1000 seeds and for threshold values $T = 0.01$ and $T = 0.1$. Finally, we show the total running time of our algorithm when it is stopped after computing an approximate greedy sequence of 50, 1000, or *all n nodes*. Note that we omit the respective influence figure for the seed set that contains all nodes, since it is 100% by definition.

The table shows that, unsurprisingly, the higher threshold has higher influence values. This is because the coverage function is monotone non-decreasing in T. The running time of our algorithm depends on that influence (since its graph searches must run for longer), but it is still practical even for fairly large thresholds and even if we compute the entire permutation. For the largest graph we test (Orkut), with hundreds of millions of edges, we can compute the top 50 seeds in less than 15 minutes, and order all nodes in a few hours using a single CPU core.

Figure 4 presents a more detailed perspective on the same experiment. It shows, for $T = 0.01$ and $T = 0.1$, how total influence

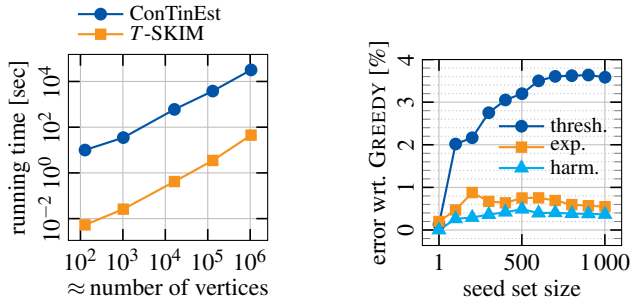

Figure 1: Left: Comparing T-SKIM to ConTinEst. Right: Error of T-SKIM and α-SKIM.

Table 1: Performance of α-SKIM using $k = 64$, $\ell = 64$, and exponentially distributed edge weights for 50 and 1000 seeds. We use exponential (exp.: $\alpha\colon x \mapsto e^{-10x}$) and harmonic (harm.: $\alpha\colon x \mapsto 1/(10x+1)$) decay functions.

	INFLUENCE [%]				RUNNING TIME [SEC]			
	50 seeds		1000 seeds		50 seeds		1000 seeds	
instance	exp.	harm.	exp.	harm.	exp.	harm.	exp.	harm.
AstroPh	17.6	31.4	33.5	44.9	15	15	43	40
Epinions	7.6	14.9	11.2	18.2	35	40	93	99
Slashdot	16.9	29.1	21.3	32.8	104	88	238	224
Gowalla	13.1	25.9	15.9	28.2	166	213	323	455
TwitterF's	16.0	26.3	19.7	29.2	1,500	1,387	2,459	2,816
LiveJournal	10.6	23.5	13.4	25.8	5,637	7,765	11,906	13,016

and the running times depend on the size of the seed set. We note that the first few seeds contribute with a disproportionate fraction of the total influence, particularly with $T = 0.1$, and an even higher percentage of the total running time. The overall shape of the curves is quite similar, with Orkut as a noticeable outlier: its first few seeds contribute relatively more to the overall influence than in other instances. Note that Orkut is also the densest instance in our testbed.

We now compare T-SKIM to ConTinEst, the algorithm by Du et al. [18]. Although their sequential implementation is publicly available, we were unable to run it on our inputs within reasonable time. (A preliminary test on AstroPh, our smallest instance, did not produce any output within five hours.) Note that to evaluate graphs with more than 1024 vertices, they actually use a distributed implementation, which they run on a cluster with 192 cores. Unfortunately, we had access neither to such a cluster nor to the distributed implementation of their algorithm.

In order to still be able to make some comparison, we generated the same instances as they used in their evaluation: core-periphery Kronecker networks [33] (parameter matrix: [0.9 0.5; 0.5 0.3]) of varying size, using the Weibull distribution for the edge lengths [26]. (Note that λ controls scale and β shape.) For each edge we chose λ and β uniformly at random from $(0, 10]$. We ran the same experiment as they did, setting $|S| = 10$, and $T = 10$. Figure 1 (left) shows the running times for Kronecker networks of varying size. We observe that our approach run on a single CPU core is consistently about 3 orders of magnitude faster than their algorithm run on a cluster. Unfortunately, we were not able to compare the computed influence, as those figures are not reported in [18].

5.1.2 General Distance-Based Influence

We now evaluate α-SKIM, a more general version of our IM algorithm that can handle arbitrary decay functions. For this experiment, we consider both harmonic and exponential decay functions, the most commonly used in the literature. To test harmonic decay, we use $\alpha(x) = 1/(10x+1)$; for exponential decay, we use $\alpha = e^{-10x}$. These functions turn out to give interesting influence profiles. In α-SKIM we initialize τ to $n\ell/k$ and set λ to 0.5.

Table 1 shows, for both functions, the influence values (in percent) obtained by α-SKIM for 50 and 1000 seeds, as well as the corresponding running times.

The table shows that α-SKIM is slower than T-SKIM by up to an order of magnitude for comparable influence. In fact, if we ran α-SKIM with a threshold function (not shown in the table), it would be about three times as slow as T-SKIM, while producing the exact same results. However, this is to be expected, since α-SKIM is a much more sophisticated (and flexible) algorithm, which, unlike T-SKIM, can handle smooth decay functions with guarantees.

Even though α-SKIM is slower, it is still practical. It scales well with the number of seeds (increasing from 50 to 1000 barely doubles the total running time) and can still handle very large graphs.

Figure 2 presents a more detailed view of the same experiment (for a few graphs), with up to n seeds. It shows that computing a full permutation (with n seeds) is not much more expensive than computing $n/1000$ (a few dozen) seeds. An interesting difference between these results and those for T-SKIM (reported in Figure 4) is that for α-SKIM the running time grows less smoothly with the number of seeds. The discontinuities correspond to decreases in the sampling threshold τ, causing additional sampling.

5.1.3 Solution Quality

Figure 1 (right) compares the quality of the seed sets found by T-SKIM (for threshold decay) and α-SKIM (for exponential and harmonic decays) with those found by exact GREEDY on AstroPh ($\ell = 64$ simulations). We consider sets of size 1 to 10^3 and the same decay functions as above. Each point of the curve represents the error (in percent) of our algorithm when compared to GREEDY. We observe that the error is very low in general (less than 1% for exponential and harmonic decay, and less than 4% for threshold). Considering the fact that SKIM is many orders of magnitude faster than GREEDY (while still providing strong guarantees), these errors are acceptable. Note that the error of the first seed vertex is very low in all cases (close to 0%), indicating that SKIM does very well in finding the most influential node.

The quality of the solutions provided by the algorithm with respect to the probabilistic input (graph distribution) depends on the number of instances (simulations) ℓ. Our experiments so far have used $\ell = 64$. We now compare this with other choices of ℓ. Figure 3 compares the quality of the seed sets found by GREEDY for AstroPh for $\ell = 4, 16, 64, 128$ with those found by $\ell = 256$. We consider sets of size 1 to 50 and three different decay functions: exponential, harmonic, and threshold (with $T = 0.01$). Each point in the curve represents the error (in percent) relative to the solution with $\ell = 256$. Although the error is consistently high for the threshold IM when ℓ is very small, it becomes negligible for $\ell \geq 64$, justifying our choice of parameters. For smoother (exponential or harmonic) decay, all errors are significantly smaller, and even smaller values of ℓ would be acceptable.

5.2 Distance-Based Influence Oracle

We now evaluate our influence oracles. Recall that this setting has two stages. The preprocessing stage takes as input only the graph and computes sketches. The query stage takes a set S of seeds and a function α and uses the sketches to estimate the influence of S with respect to α. Note that same preprocessing stage can be

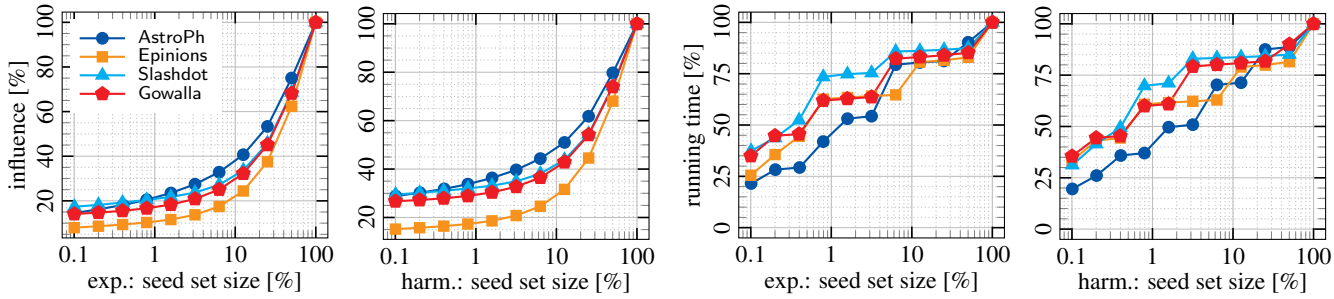

Figure 2: Evaluating influence permutations (left) and running time (right) on several instances for exponential (exp.: $\alpha\colon x \mapsto e^{-10x}$) and harmonic (harm.: $\alpha\colon x \mapsto\colon 1/(10x+1)$) decays. The legend applies to all plots.

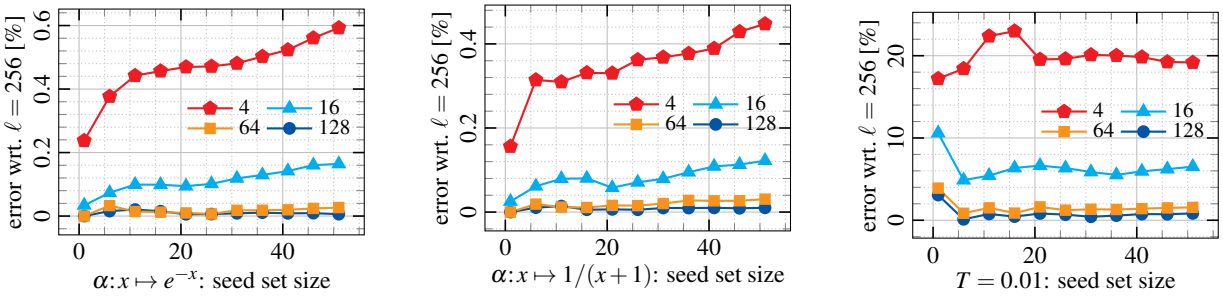

Figure 3: Evaluating different numbers of simulations (ℓ-values) for different decay functions on AstroPh.

Table 2: Evaluating the distance-based influence oracle with $\ell = 64$.

	Preprocessing		Queries with $\alpha\colon x \mapsto e^{-10x}$						Queries with $\alpha\colon x \mapsto 1/(10x+1)$						Queries with $T = 0.01$					
			1 seed		50 seeds		1000 seeds		1 seed		50 seeds		1000 seeds		1 seed		50 seeds		1000 seeds	
instance	time [h:m]	space [MiB]	time [µs]	err. [%]	time [µs]	err. [%]	time [µs]	err. [%]	time [µs]	err. [%]	time [µs]	err. [%]	time [µs]	err. [%]	time [µs]	err. [%]	time [µs]	err. [%]	time [µs]	err. [%]
AstroPh	0:10	149.2	38	7.2	9,695	1.2	229,340	0.5	31	4.4	9,152	4.1	227,943	0.5	27	1.1	8,855	0.4	204,551	2.8
Epinions	0:46	674.0	32	3.2	8,552	1.1	222,470	1.0	26	2.2	9,203	1.2	196,717	0.5	22	0.5	8,267	0.3	191,709	0.6
Slashdot	1:10	851.4	46	5.6	11,884	1.5	310,170	0.4	38	3.2	10,970	1.9	291,185	1.2	73	0.6	13,768	0.4	247,509	0.6
Gowalla	3:55	2,558.6	52	3.8	17,109	1.0	356,818	0.4	47	2.9	14,151	2.2	289,318	0.8	61	1.2	16,092	0.6	329,976	0.3
TwitterF's	19:33	6,165.1	51	3.8	13,816	1.4	365,366	0.7	42	2.6	13,166	1.5	379,296	0.9	39	2.3	13,912	0.7	360,766	0.2

used to answer queries for any decay function α. For this experiment, we consider three such functions: exponential ($\alpha(x) = e^{-10x}$), harmonic ($\alpha(x) = 1/(10x+1)$), and threshold (with $T = 0.01$).

Table 2 summarizes our results in this setting. For each dataset tested, it first shows the preprocessing time and the total space required to store all sketches. Then, for each decay function, we report the query time (in microseconds) and the estimation error for random sets S of sizes 1, 50, and 1000. (Note that measuring the error requires computing exact influence of each seed set with multiple Dijkstra searches; this time is not included in the table.) Each entry in the table is the average of 100 random seed sets.

The table shows that, as predicted, query times are almost independent of the α function, the size of the influenced set, and the size of the graph. Moreover, they have a slightly superlinear dependence on the number of seeds. Queries are somewhat slower than for reachability-based IC (as reported in [12]), since sketches are bigger and the estimator is more involved. Our oracles are much more flexible, however, and still practical. For 50 seeds, one can answer queries in a few milliseconds, whereas an exact computation could take minutes or more on large graphs. Moreover, its error is consistently low, regardless of the number of seeds.

Acknowledgements. We would like to thank the authors of [18] for helping us reproduce their inputs and pointing us to their implementation of ConTinEst.

6. REFERENCES

[1] B. D. Abrahao, F. Chierichetti, R. Kleinberg, and A. Panconesi. Trace complexity of network inference. In *KDD*, 2013.

[2] A. Bavelas. A mathematical model for small group structures. *Human Organization*, 7:16–30, 1948.

[3] F. Bloch and M. O. Jackson. The formation of networks with transfers among players. *Journal of Economic Theory*, 133(1):83–110, 2007.

[4] P. Boldi and S. Vigna. In-core computation of geometric centralities with hyperball: A hundred billion nodes and beyond. In *ICDM workshops*, 2013. `http://arxiv.org/abs/1308.2144`.

[5] P. Boldi and S. Vigna. Axioms for centrality. *Internet Mathematics*, 2014.

[6] C. Borg, M. Brautbar, J. Chayes, and B. Lucier. Maximizing social influence in nearly optimal time. In *SODA*, 2014.

[7] W. Chen, Y. Wang, and S. Yang. Efficient influence maximization in social networks. In *KDD*. ACM, 2009.

[8] E. Cohen. Size-estimation framework with applications to transitive closure and reachability. *J. Comput. System Sci.*, 55:441–453, 1997.

[9] E. Cohen. All-distances sketches, revisited: HIP estimators for massive graphs analysis. In *PODS*. ACM, 2014.

[10] E. Cohen. Estimation for monotone sampling: Competitiveness and customization. In *PODC*. ACM, 2014. full version

Table 3: Performance of T-SKIM using $k = 64$, $\ell = 64$, and exponentially distributed edge weights. We evaluate the influence on 512 (different) sampled instances for thresholds 0.1 and 0.01.

			Influence [%]				Running Time [sec]					
			50 seeds		1000 seeds		50 seeds		1000 seeds		n seeds	
instance	# nodes	# edges	0.01	0.1	0.01	0.1	0.01	0.1	0.01	0.1	0.01	0.1
AstroPh	14,845	239,304	1.02	19.17	9.96	39.25	0.9	2.0	2.0	4.0	3.7	6.6
Epinions	75,888	508,837	0.53	8.52	2.88	12.68	2.0	5.2	6.3	11.1	14.1	21.3
Slashdot	77,360	828,161	0.72	19.97	3.90	25.04	1.9	14.6	7.6	27.9	18.9	40.5
Gowalla	196,591	1,900,654	0.62	14.13	1.93	17.61	4.4	21.8	14.8	36.9	47.6	81.7
TwitterF's	456,631	14,855,852	0.20	19.38	1.64	24.26	9.9	133.4	36.4	269.6	269.9	648.4
LiveJournal	4,847,571	68,475,391	0.07	9.16	0.33	13.81	34.6	606.0	117.5	1,244.4	1,983.4	4,553.9
Orkut	3,072,627	234,370,166	2.82	74.44	4.61	77.47	779.7	5,490.5	1,788.7	11,060.7	7,360.9	24,520.3

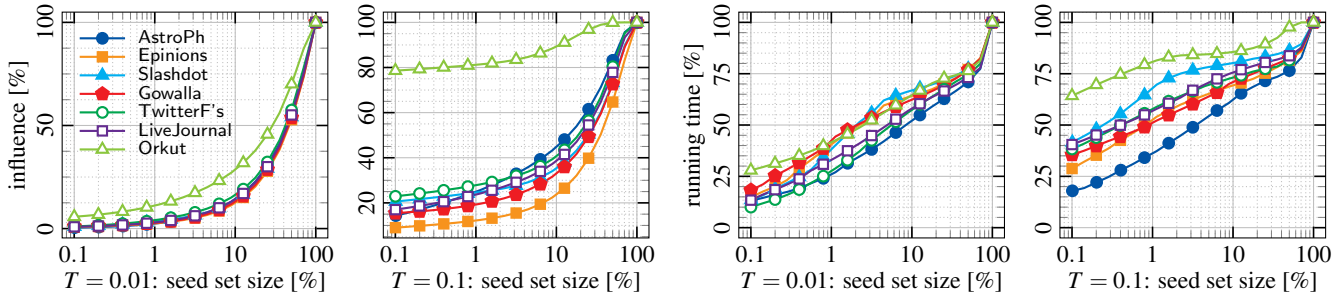

Figure 4: Evaluating influence permutations (left) and running times (right) on several instances for threshold decays 0.01 and 0.1. The legend applies to all plots.

`http://arxiv.org/abs/1212.0243`.

[11] E. Cohen, D. Delling, F. Fuchs, A. Goldberg, M. Goldszmidt, and R. Werneck. Scalable similarity estimation in social networks: Closeness, node labels, and random edge lengths. In *COSN*. ACM, 2013.

[12] E. Cohen, D. Delling, T. Pajor, and R. F. Werneck. Sketch-based influence maximization and computation: Scaling up with guarantees. In *CIKM*. ACM, 2014.

[13] E. Cohen and H. Kaplan. Spatially-decaying aggregation over a network: Model and algorithms. *J. Comput. System Sci.*, 73:265–288, 2007. Full version of a SIGMOD 2004 paper.

[14] E. Cohen and H. Kaplan. Summarizing data using bottom-k sketches. In *ACM PODC*, 2007.

[15] E. Cohen and H. Kaplan. Leveraging discarded samples for tighter estimation of multiple-set aggregates. In *ACM SIGMETRICS*, 2009.

[16] E. Cohen and M. Strauss. Maintaining time-decaying stream aggregates. *J. Algorithms*, 59:19–36, 2006.

[17] Ch. Dangalchev. Residual closeness in networks. *Phisica A*, 365, 2006.

[18] N. Du, L. Song, M. Gomez-Rodriguez, and H. Zha. Scalable influence estimation in continuous-time diffusion networks. In *NIPS*. Curran Associates, Inc., 2013.

[19] U. Feige. A threshold of $\ln n$ for approximating set cover. *J. Assoc. Comput. Mach.*, 45:634–652, 1998.

[20] L. C. Freeman. Centrality in social networks: Conceptual clarification. *Social Networks*, 1, 1979.

[21] J. Goldenberg, B. Libai, and E. Muller. Talk of the network: A complex systems look at the underlying process of word-of-mouth. *Marketing Letters*, 12(3), 2001.

[22] M. Gomez-Rodriguez, D. Balduzzi, and B. Schölkopf. Uncovering the temporal dynamics of diffusion networks. In *ICML*, 2011.

[23] M. O. Jackson. *Social and economic networks*. Princeton University Press, 2010.

[24] K. Jung, W. Heo, and W. Chen. Irie: Scalable and robust influence maximization in social networks. In *ICDM*. ACM, 2012.

[25] D. Kempe, J. M. Kleinberg, and É. Tardos. Maximizing the spread of influence through a social network. In *KDD*. ACM, 2003.

[26] J. F. Lawless. *Statistical models and methods for lifetime data*, volume 362. John Wiley & Sons, 2011.

[27] J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. VanBriesen, and Glance N. Cost-effective outbreak detection in networks. In *KDD*. ACM, 2007.

[28] G. Nemhauser, L. Wolsey, and M. Fisher. An analysis of the approximations of maximizing submodular set functions. *Mathematical Programming*, 14, 1978.

[29] T. Opsahl, F. Agneessens, and J. Skvoretz. Node centrality in weighted networks: Generalizing degree and shortest paths. *Social Networks*, 32, 2010. `http://toreopsahl.com/2010/03/20/`.

[30] M. Richardson and P. Domingos. Mining knowledge-sharing sites for viral marketing. In *KDD*. ACM, 2002.

[31] M. Rosenblatt. Remarks on some nonparametric estimates of a density function. *The Annals of Mathematical Statistics*, 27(3):832, 1956.

[32] G. Sabidussi. The centrality index of a graph. *Psychometrika*, 31(4):581–603, 1966.

[33] Stanford network analysis project. `http://snap.stanford.edu`.

[34] Y. Tang, Y. Shi, and X. Xiao. Influence maximization in near-linear time: A martingale approach. In *SIGMOD*, 2015.

[35] Y. Tang, X. Xiao, and Y. Shi. Influence maximization: Near-optimal time complexity meets practical efficiency. In *SIGMOD*, 2014.