Inferring Networks of Diffusion and Influence

Manuel Gomez-Rodriguez
Stanford University
MPI for Biological Cybernetics
manuelgr@stanford.edu

Jure Leskovec
Stanford University
jure@stanford.edu

Andreas Krause
California Institute of Technology
krausea@caltech.edu

ABSTRACT

Information diffusion and virus propagation are fundamental processes talking place in networks. While it is often possible to directly observe when nodes become infected, observing individual transmissions (i.e., who infects whom or who influences whom) is typically very difficult. Furthermore, in many applications, the underlying network over which the diffusions and propagations spread is actually *unobserved*. We tackle these challenges by developing a method for tracing paths of diffusion and influence through networks and inferring the networks over which contagions propagate. Given the times when nodes adopt pieces of information or become infected, we identify the optimal network that best explains the observed infection times. Since the optimization problem is NP-hard to solve exactly, we develop an efficient approximation algorithm that scales to large datasets and in practice gives provably near-optimal performance.

We demonstrate the effectiveness of our approach by tracing information cascades in a set of 170 million blogs and news articles over a one year period to infer how information flows through the online media space. We find that the diffusion network of news tends to have a core-periphery structure with a small set of core media sites that diffuse information to the rest of the Web. These sites tend to have stable circles of influence with more general news media sites acting as connectors between them.

Categories and Subject Descriptors: H.2.8 **[Database Management]**: Database applications—*Data mining*
General Terms: Algorithms; Experimentation.
Keywords: Networks of diffusion, Information cascades, Blogs, News media, Meme-tracking, Social networks.

1. INTRODUCTION

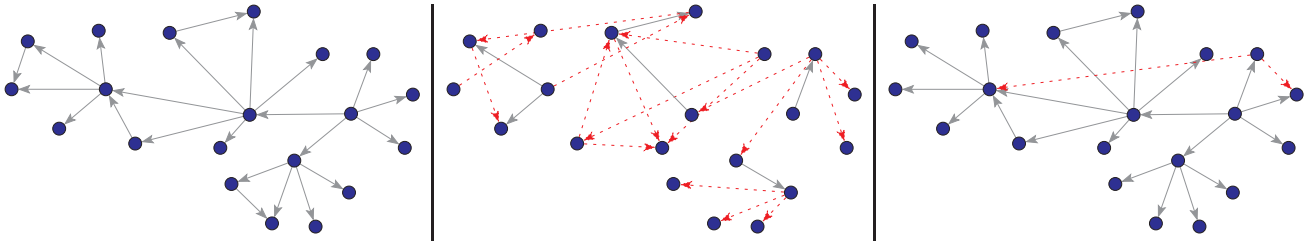
Cascading behavior, diffusion and spreading of ideas, innovation, information, influence, viruses and diseases are fundamental processes taking place in networks [12, 28, 30]. In order to study network diffusion there are two fundamental challenges one has to address. First, to be able to track cascading processes taking place in a network, one needs to identify the contagion (idea, information, virus, disease) that is actually spreading and propagating over the edges of the network. Moreover, one has to then identify a way to successfully trace the contagion. For example, when tracing information diffusion, it is a non-trivial task to automatically and on a large scale identify the phrases or "memes" that are spreading through the Web. Second, we usually think of diffusion as a process that takes place on a *network*. However, the network over which propagations take place is usually *unknown* and *unobserved*. Commonly, we only observe the times when particular nodes get "infected" but we *do not* observe *who* infected them. In case of information propagation, as bloggers discover new information, they write about it without explicitly citing the source. Thus, we only observe the time when a blog gets "infected" but not where it got infected from. Similarly, in virus propagation, we observe people getting sick without usually knowing who infected them. And, in a viral marketing setting, we observe people purchasing products or adopting particular behaviors without explicitly knowing who was the influencer that caused the adoption or the purchase.

These challenges are especially pronounced in information diffusion on the Web, where there have been relatively few large scale studies of information propagation in large networks [2, 23, 24, 25]. In order to study paths of diffusion over networks, one essentially requires to have complete information about who influences whom, as a single missing link in a sequence of propagations can lead to wrong inferences. Even if one collects near complete large scale diffusion data, it is a non-trivial task to identify textual fragments that propagate relatively intact through the Web without human supervision. And even then the question of how information diffuses through the network still remains. Thus, the questions are, what is the network over which the information propagates on the Web? What is the global structure of such a network? How do news media sites and blogs interact? What roles do different sites play in the diffusion process and how influential are they?

Our approach to inferring networks of diffusion and influence. We address the above questions by positing that there is some underlying unknown network over which information, viruses or influence propagate. For simplicity, we assume that the underlying network is static and does not change over time. We then observe the times when nodes get "infected" by a particular contagion (a particular piece of information, product or virus) but we do not observe where they got infected from. Thus, for each cascade, we only observe times when nodes got infected, and we are then interested in determining the paths the diffusion took through the unobserved network. Our goal is to reconstruct the network over which cascades propagate.

Edges in such networks of influence and diffusion have various interpretations. In virus or disease propagation, edges can be interpreted as who-infects-whom. In information propagation, edges are

KDD'10, July 25–28, 2010, Washington, DC, USA.



(a) True diffusion network (b) Inferred network using heuristic (c) Inferred network using NETINF

Figure 1: Diffusion network inference problem. There is an unknown network (a) over which cascades spread. Using a baseline heuristic (see Section 4) we recover network (b), while our method (c) almost perfectly recovers the network.

who-copies-from-whom or who-listens-to-whom. In a viral marketing setting, edges can be understood as who-influences-whom.

The main premise of our work is that by observing many different cascades spreading among the nodes, we can infer the edges of the underlying propagation network. If node v tends to get infected soon after node u for many different cascades, then we can expect an edge (u, v) to be present in the network. By exploring correlations in node infection times, we aim to recover the unobserved diffusion network.

Here we develop NETINF, a scalable algorithm for inferring networks of diffusion and influence. We first formulate a generative probabilistic model of how, on a fixed hypothetical network, cascades spread as directed trees through the network. Since we only observe times when nodes get infected, there are many possible propagation trees that explain the same data and we have to consider all of them. Thus, naive computation of the model takes exponential time since there is a combinatorially large number of propagation trees. We show that, perhaps surprisingly, computations over this super-exponential set of trees can be performed in cubic time. However, under such model, the network inference problem is still intractable. Thus, we introduce a tractable approximation, and show that the resulting objective function can be both efficiently computed and efficiently optimized. By exploiting a diminishing returns property of the model, we prove that NETINF infers near-optimal networks. We also speed-up NETINF algorithm by exploiting the local structure of the objective function and by using lazy evaluations [21].

Our results on synthetic datasets show that we can reliably infer the underlying propagation and influence networks, regardless of the overall network structure. Validation on real and synthetic datasets shows that NETINF outperforms a baseline heuristic by an order of magnitude and correctly discovers more than 90% of the edges. We apply our algorithm to a real Web information propagation dataset of 170 million blog and news articles over a one year period. Our results show that online news propagation networks tend to have a core-periphery structure with a small set of core blog and news media websites that diffuse information to the rest of the Web, news media websites tend to diffuse the news faster than blogs and blogs keep discussing about news longer time than media websites.

Inferring how information or viruses propagate over networks is crucial for a better understanding of diffusion in networks. By modeling the structure of the propagation network, we can gain insight into positions and roles various nodes play in the diffusion process and assess the range of influence of nodes in the network.

2. PROBLEM FORMULATION

We now formally describe the problem where many different cascades propagate over an unknown static network and for each of them we observe times *when* nodes got infected but not *who* infected them. The goal then is to infer the unknown network over which cascades originally propagated. In the information diffusion setting, each cascade corresponds to a different piece of information that spreads over the network and all we observe are the times when particular sites mentioned particular information. The task then is to infer the network where a directed edge (u, v) means that a site v tends to repeat or to mention stories after a site u.

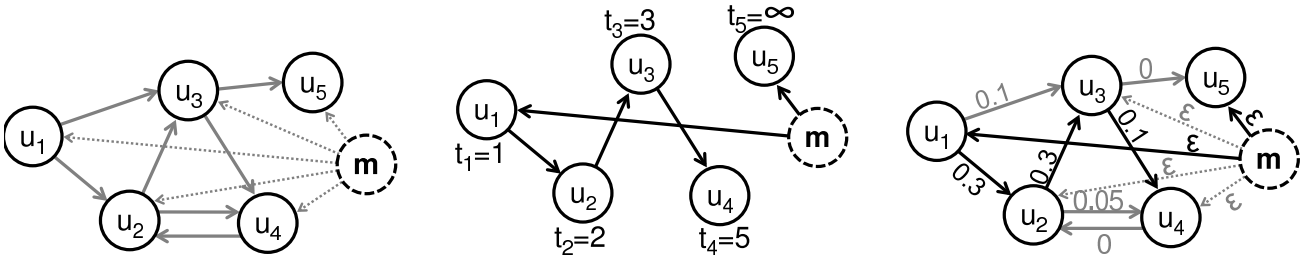
2.1 Problem statement

Given a hidden directed network G^*, we observe multiple cascades spreading over it. As the cascade c propagates over the network, it leaves a trace in the form of $(u_i, t_i, \phi_i)_c$, which means that cascade c reached node u_i at time t_i with a set of features or attributes ϕ_i. Note that we only observe the time t_u when cascade c reached node u but not how it reached the node u. Now, given such traces of many different cascades, we aim to infer the unobserved directed network G^*, i.e., the network over which the cascades originally propagated. We refer to the estimated version of the network as $\hat{G}$. We use the term *hit time* t_u to refer to the time when a cascade hits (infects) a particular node u. Many cascades will not hit all the nodes – if a node u is not hit by a cascade, then $t_u = \infty$. Thus, a cascade is fully specified by the vector $\mathbf{t} = [t_1, \ldots, t_n]$ of hit times, and the feature vector $\mathbf{\Phi} = [\phi_1, \ldots, \phi_n]$ describing the properties of the contagion and the node (where n is the number of nodes in G). Note that the probability of propagation may be a complicated function of the properties of the nodes (e.g., age, gender) and the properties of the contagion itself (e.g., product category, price). One can use the feature vector $\mathbf{\Phi}$ to describe such properties of individual nodes and contagions.

2.2 Model formulation

Suppose that for a fixed cascade $c = (\mathbf{t}, \mathbf{\Phi})$, we know which nodes influenced which other nodes. We assume that every node v in a cascade is influenced by at most one node u. Thus, the influence structure of a cascade c is given by a directed tree T, which we assume to be contained in the directed graph G, i.e., the graph over which the cascade propagated. First, we will specify the cascade transmission model $P_c(u, v)$ that describes how likely is that node u spreads the cascade c to node v. We will then describe the probability $P(c|T)$ that the cascade c propagates in a particular tree pattern T, where tree T simply specifies which nodes influence which other nodes. Last, we will define $P(c|G)$, which is the probability that cascade c occurs in a network G.

Cascade transmission model. We build on the independent cascade model [13] which posits that an infected node infects each of its neighbors independently with some chosen probability. As in this model the time is modeled only implicitly through the epochs of the propagation we fist extend the independent cascade model to continuous time domain. We account for the fact that information can spread quickly over some edges while over others it may take



(a) Network G (b) Cascade c on G (c) Max spanning tree

Figure 2: (a) Diffusion network G. (b) A cascade c on nodes $u_1, \ldots, u_5$ with infection times t_u, and most likely propagation tree (black edges). As node u_1 does not have a parent, the ε-edge (m, u_1) is picked. (c) The maximum directed spanning tree of a graph is obtained by each node picking an incoming edge of maximum weight.

much longer to propagate. Also note that cascades in such a model are necessarily trees since if a node u gets infected multiple times knowing the node that infected u first is sufficient.

First we define the probability of observing a particular cascade $c = (\mathbf{t}, \mathbf{\Phi})$ of hit times and features for a fixed propagation tree T. Consider a single edge $(u, v) \in T$. Given the hit times $(u, t_u)_c$ and $(v, t_v)_c$ $(t_u < t_v)$ of nodes u and v in cascade c, we aim to estimate the probability $P_c(u, v)$ that a node u spreads the cascade to a node v (i.e. a node u influences/infects/transmits to a node v) in the tree T. Formally, $P_c(u, v)$ specifies the conditional probability of observing cascade c spreading from node u to node v.

Influence can only propagate forward in time. Thus, if $t_u \geq t_v$, we simply set $P_c(u, v) = 0$. Generally the probability of propagation $P_c(u, v)$ between a pair of nodes u and v is decreasing in the difference of their infection times, i.e., the farther apart in time the two nodes get infected the less likely they are to infect one another. However, note that we can make the cascade transmission model $P_c(u, v)$ arbitrarily complicated as it can also depend on feature vectors ϕ_u and ϕ_v that describe the properties of the contagion and the properties of the nodes. For example, in a disease propagation scenario, ϕ could include information about the individual's socio-economic status, commute patterns, disease history and so on. This allows for much more realistic cascade transmission models as the probability of infection depends on the parameters of the disease and properties of the nodes. For simplicity, in the rest of the paper and in all our experiments, we ignore the features and assume that the probability of transmission depends only on the time difference between the node hit times $\Delta = t_v - t_u$.

Considering the model in a generative sense, the cascade c reaches node u at time t_u, and we now need to generate the time t_v when u spreads the cascade to node v. As cascades generally do not infect all the nodes of the network, we need to explicitly model the probability that the cascade stops. With probability $(1 - \beta)$, the cascade stops, and never reaches v, thus $t_v = \infty$. With probability β, the cascade continues, and the hit time t_v is set to $t_u + \Delta$, where Δ is the waiting time that passed between the hit times t_u and t_v. We consider two different models for the waiting time Δ, an exponential and a power-law model, each with parameter α:

$$P(\Delta) \propto e^{-\frac{\Delta}{\alpha}} \text{ and } P(\Delta) \propto \frac{1}{\Delta^{\alpha}}.$$

We consider both the power-law and exponential waiting time models since they have both been argued for in the literature [3, 23, 26]. In the end, our algorithm does not depend on the particular choice of the waiting time distribution and more complicated functions can easily be chosen [6]. Also, we interpret $\infty + \Delta = \infty$, i.e., if $t_u = \infty$, then $t_v = \infty$ with probability 1.

Now that we specified the probability $P_c(u, v)$ that node u influences node v, we define the probability of observing cascade c propagating in a particular tree structure T as

$$P(c|T) = \prod_{(i,j)\in T} P_c(i, j),$$

where the product is over the edges of the tree T. Here, the edges of the tree T simply specify how the cascade spreads, i.e., every node gets influenced by its parent.

Cascade propagation model. We just defined the probability of a single cascade c propagating in a particular tree pattern T, $P(c|T)$. Now, our aim is to compute $P(c|G)$, the probability that a cascade c occurs in a graph G. Note that cascade is defined only by the node infection times and the propagation tree T (who-infected-whom) is unknown. So, we combine the probabilities of individual propagation trees into a probability of a cascade c occurring in a network G. We do this by considering all possible propagation trees T, i.e., all possible ways in which cascade c could have spread over G:

$$P(c|G) = \sum_{T\in\mathcal{T}(G)} P(c|T)P(T|G) \propto \sum_{T\in\mathcal{T}(G)} \prod_{(i,j)\in T} P_c(i, j) \tag{1}$$

where c is a cascade and $\mathcal{T}(G)$ is the set of all the directed spanning trees on G. Basically, the graph G defines the skeleton over which the cascades propagate and T defines a particular possible propagation. Since we do not know in which particular tree pattern the cascade really propagated, we consider all possible propagation trees T in $\mathcal{T}(G)$. Thus, the sum over T is a sum over all directed spanning trees in $\mathcal{T}(G)$. We assume that all propagation trees are a priori equally likely, i.e., $P(T|G)$ is the uniform distribution over all directed spanning trees.

We just computed the probability of a single cascade c occurring in G, and we now define the probability of a set of cascades C occurring in G simply as

$$P(C|G) = \prod_{c\in C} P(c|G), \tag{2}$$

where we assume conditional independence between cascades given the graph G.

Network inference problem. Next, we define the *diffusion network inference problem*, where we aim to find $\hat{G}$ that solves the following optimization problem:

$$\hat{G} = \underset{|G|\leq k}{\operatorname{argmax}}\, P(C|G),$$

where the maximization is over all graphs G of at most k edges. We add the constraint on the number of edges in $\hat{G}$ for two reasons.

First, the optimization problem without the constraint would have a trivial solution since the fully connected graph maximizes the above quantity. Second, since real graphs are sparse, we also seek for a sparse solution. We discuss how to choose k later.

The above optimization problem seems wildly intractable. To evaluate Eq. (2), we need to compute Eq. (1) for each cascade c, i.e., the sum over all possible spanning trees T. The number of trees can be super-exponential in the size of G but perhaps surprisingly, this super-exponential sum can be performed in time polynomial in the number n of nodes in the graph G, by applying Kirchhoff's matrix tree theorem [15]:

THEOREM 1 (TUTTE (1948)). *If we construct a matrix A such that $a_{i,j} = \sum w_{k,j}$ if $i = j$ and $a_{i,j} = -w_{i,j}$ if $i \neq j$ and if $A_{k,m}$ is the matrix created by removing any row k and column m from A such that $k + m$ is an even number, then*

$$\det(A_{k,m}) = \sum_{T \in A} \prod_{(i,j) \in T} w_{i,j}, \tag{3}$$

where T is each directed spanning tree in A.

In our case, we set $w_{i,j}$ to be simply $P_c(i,j)$ and we compute the product of the determinants of $|C|$ matrices, one for each cascade, which is exactly Equation 1. This means that instead of using super-exponential time, we are now able to evaluate Eq. 2 in time $O(|C| \cdot n^3)$ (the time required to compute $|C|$ determinants). However, this does not completely solve our problem for two reasons: First, while cubic time is a drastic improvement over a super-exponential computation, it is still too expensive for the large graphs that we want to consider. Second, we can use the above result only to evaluate the quality of *a particular* graph G, while our goal is to find the *best* graph $\hat{G}$. To do this, one would need to search over *all* graphs G to find the best one. Again, as there is a super-exponential number of graphs, this is practically impossible. One could propose some search heuristics, like hill-climbing, however, due to the combinatorial nature of the likelihood function, such a procedure would likely be prone to local minima.

3. PROPOSED ALGORITHM

The diffusion network inference problem defined in the previous section does not allow for an efficient solution. We now propose an alternative formulation of the problem that is tractable to compute *and* to optimize.

3.1 An alternative formulation

For each cascade c, instead of considering every possible propagation (spanning) tree T, we consider only the most likely propagation tree T:

$$P(C|G) = \prod_{c \in C} \max_{T \in \mathcal{T}(G)} P(c|T) = \prod_{c \in C} \max_{T \in \mathcal{T}(G)} \prod_{(i,j) \in T} P_c(i,j). \tag{4}$$

We then define the improvement of log-likelihood for cascade c under graph G over an empty graph $\bar{K}$:

$$F_c(G) = \max_{T \in \mathcal{T}(G)} \log P(c|T) - \max_{T \in \mathcal{T}(\bar{K})} \log P(c|T). \tag{5}$$

Note that maximizing Eq. (4) is equivalent to maximizing the following objective function:

$$F_C(G) = \sum_{c \in C} F_c(G)$$

In reality, nodes may get infected for reasons other than the network influence. For example, in online media, not all the information propagates via the network, as some is also pushed onto the network by the mass media [12, 30] and thus a disconnected cascade can be created. Similarly, in viral marketing, a person may purchase a product due to the influence of peers (i.e., network effect) or for some other reason (e.g., seing a commercial on TV) [17].

In order to account for such phenomena when a cascade "jumps" across the network, we introduce an additional node m that represents an external source that can infect *any* node u. We connect the external influence source m (i.e., the mass media node) to every other node u with an ε-edge. And then every node u can get infected by the external source m with a very small probability ε.

Putting it all together, we include the additional node m in every cascade c and we set the probability of a cascade spreading from m to any node j in the cascade c to $P_c(m,j) = \varepsilon$. Given that we are accounting for reasons other than the network influence for a node to get infected, we assume that the ε-edges between m and all nodes in the cascade c exist also for the empty graph $\bar{K}$. We now expand Eq. (5) as

$$F_c(G) = \max_{T \in \mathcal{T}(G)} \sum_{(i,j) \in T} w_c(i,j),$$

where $w_c(i,j) = \log P_c(i,j) - \log \varepsilon$ is a non-negative weight which can be interpreted as the improvement in log-likelihood of edge (i,j) under the most likely spanning tree T in G. This means that the most likely propagation tree T is simply the *maximum weight directed spanning tree* in graph G, where each edge (i,j) has weight $w_c(i,j)$, and $F_c(G)$ is simply the sum of the weights of edges in T. Figures 2(a) and 2(b) illustrate the notion of a cascade on a directed graph, as well as the concept of ε-edges. Note that since edges (i,j) where $t_i \geq t_j$ have weight 0 (i.e., they are not present) and the node m has only outgoing edges, for a fixed cascade c, the collection of edges with positive weight forms a directed *acyclic* graph (DAG). Interestingly, for such a DAG, the maximum weight directed spanning tree can be computed efficiently:

PROPOSITION 1. *In a DAG G with vertex set V and nonnegative edge weights w, the maximum weight directed spanning tree can be found by choosing, for each node v, an incoming edge (u,v) with maximum weight $w(u,v)$.*

PROOF. The score

$$S(T) = \sum_{(i,j) \in T} w(i,j) = \sum_{i \in V} w(Par_T(i), i)$$

of a tree T is the sum of the incoming edge weights $w(Par_T(i), i)$ for each node i, where $Par_T(i)$ is the parent of node i in T (and the root is handled appropriately). Now,

$$\max_T S(T) = \max_T \sum_{(i,j) \in T} w(i,j) = \sum_{i \in V} \max_{Par_T(i)} w(Par_T(i), i).$$

Latter equality follows from the fact that since G is a DAG, the maximization can be done independently for each node without creating any cycles. □

This proposition is a special case of the more general maximum spanning tree (MST) problem in directed graphs [7]. The important fact now is that we can find the best propagation tree T in time $O(|G|)$ linear in the number of edges by simply selecting an incoming edge of highest weight for each node (Figure 2(c)).

3.2 Efficient optimization

By construction $F_C(\bar{K}) = 0$, i.e., the empty graph has score 0. Also note that F_C is non-negative and monotonic,

It can be seen that the objective function F_C is monotonic, i.e., $F_C(G) \leq F_C(G')$, whenever $G \subseteq G'$. Hence adding more edges to G does not decrease the solution quality, and thus the complete graph maximizes F_C. However, in practice, we are interested in inferring *sparse* graphs, that only contain a small number k of relevant edges. Thus we would like to solve

$$G^* = \operatorname*{argmax}_{|G| \leq k} F_C(G). \tag{6}$$

Naively searching over all k edge graphs would take time exponential in k, which is intractable. Moreover, finding the optimal solution to Eq. (6) is NP-hard, so we cannot expect to find the optimal solution:

THEOREM 2. *The diffusion network inference problem defined by equation* (6) *is NP-hard.*

PROOF. By reduction from the MAX-k-COVER problem [14]. In MAX-k-COVER, we are given a finite set W, $|W| = n$ and a collection of subsets $S_1, \ldots, S_m \subseteq W$. The function

$$F_{MC}(A) = |\cup_{i \in A} S_i|$$

counts the number of elements of W covered by sets indexed by A. Our goal is to pick a collection of k subsets A maximizing F_{MC}. We will produce a collection of n cascades C over a graph G such that $\max_{|G| \leq k} F_C(G) = \max_{|A| \leq k} F_{MC}(A)$. Graph G will be defined over the set of vertices $V = \{1, \ldots, m\} \cup \{r\}$, i.e., there is one vertex for each set S_i and one extra vertex r. For each element $s \in W$ we define a cascade which has time stamp 0 associated with all nodes $i \in V$ such that $s \in S_i$, time stamp 1 for node r and ∞ for all remaining nodes. Furthermore, we can choose the transmission model such that $w_c(i, r) = 1$ whenever $s \in S_i$ and $w_c(i', j') = 0$ for all remaining edges (i', j'), by choosing the parameters ε, α and β appropriately. Since a directed spanning tree over a graph G can contain at most one edge incoming to node r, its weight will be 1 if G contains any edge from a node i to r where $s \in S_i$, and 0 and otherwise. Thus, a graph G of at most k edges corresponds to a feasible solution A_G to MAX-k-COVER where we pick sets S_i whenever edge $(i, r) \in G$, and each solution A to MAX-k-COVER corresponds to a feasible solution G_A of (6). Furthermore, by construction, $F_{MC}(A_G) = F_C(G)$. Thus, if we had an efficient algorithm for deciding whether there exists a graph G, $|G| \leq k$ such that $F_C(G) > c$, we could use the algorithm to decide whether there exists a solution A to MAX-k-COVER with value at least c. □

While finding the optimal solution is hard, we will now show that F_C satisfies *submodularity*, a natural diminishing returns property, which will allow us to efficiently find a *provably near-optimal* solution to this NP-hard problem.

A set function $F : 2^W \to \mathbb{R}$ mapping subsets of a finite set W to the real numbers is *submodular* if whenever $A \subseteq B \subseteq W$ and $s \in W \backslash B$, it holds that $F(A \cup \{s\}) - F(A) \geq F(B \cup \{s\}) - F(B)$, i.e., adding s to the set A increases the score more than adding s to set B. We have the following result:

THEOREM 3. *Let V be a set of nodes, and C be a collection of cascades hitting the nodes V. Then $F_C(G)$ is a submodular function $F_C : 2^W \to \mathbb{R}$ defined over subsets $W \subseteq V \times V$ of directed edges.*

PROOF. Fix a cascade c, graphs $G \subseteq G'$ and an edge $e = (r, s)$ not contained in G'. We will show that $F_c(G \cup \{e\}) - F_c(G) \geq F_c(G' \cup \{e\}) - F_c(G')$. Since nonnegative linear combinations of submodular functions are submodular, the function $F_C(G) = \sum_{c \in C} F_c(G)$ is submodular as well. Let $w_{i,j}$ be the weight of edge (i, j) in $G \cup \{e\}$, and $w'_{i,j}$ be the weight in $G' \cup \{e\}$. As argued in Section 3.1, the maximum weight directed spanning tree for DAGs is obtained by assigning to each node the incoming edge with maximum weight. Let (i, s) be the edge incoming at s of maximum weight in G, and (i', s) the maximum weight incoming edge in G'. Since $G \subseteq G'$ it holds that $w_{i,s} \leq w'_{i',s}$. Furthermore, $w_{r,s} = w'_{r,s}$. Hence,

$$\begin{aligned} F_c(G \cup \{(r,s)\}) - F_c(G) &= \max(w_{i,s}, w_{r,s}) - w_{i,s} \\ &\geq \max(w'_{i',s}, w'_{r,s}) - w'_{i',s} \\ &= F_c(G' \cup \{(r,s)\}) - F_c(G'), \end{aligned}$$

proving submodularity of F_c. □

Maximizing submodular functions in general is NP-hard [14]. A commonly used heuristic is the *greedy algorithm*, which starts with an empty graph $\bar{K}$, and iteratively, in step i, adds the edge e_i which maximizes the marginal gain:

$$e_i = \operatorname*{argmax}_{e \in G \backslash G_{i-1}} F_C(G_{i-1} \cup \{e\}) - F_C(G_{i-1}). \tag{7}$$

The algorithm stops once it has selected k edges, and returns the solution $\hat{G} = \{e_1, \ldots, e_k\}$. The stopping criteria, i.e., value of k, can be based on some threshold of the marginal gain, of the number of estimated edges or another more sophisticated heuristic. Considering the hardness of the problem, we might expect the greedy algorithm to perform arbitrarily bad. However, we will see that this is not the case. A fundamental result of Nemhauser et al. [27] proves that for monotonic submodular functions, the set $\hat{G}$ returned by the greedy algorithm obtains at least a constant fraction of $(1 - 1/e) \approx 63\%$ of the optimal value achievable using k edges.

Moreover, we can acquire a tight *online* bound on the solution quality:

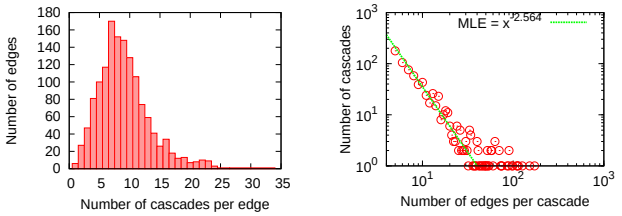
THEOREM 4 ([21]). *For a graph $\hat{G}$, and each edge $e \notin \hat{G}$, let $\delta_e = F_C(\hat{G} \cup \{e\}) - F_C(\hat{G})$. Let $e_1, \ldots e_B$ be the sequence with δ_e in decreasing order. Then,*

$$\max_{|G| \leq k} F_c(G) \leq F_c(\hat{G}) + \sum_{i=1}^{k} \delta_{e_i}.$$

Thm. 4 computes how far a given $\hat{G}$ (obtained by *any* algorithm) is from the unknown NP-hard to find optimum.

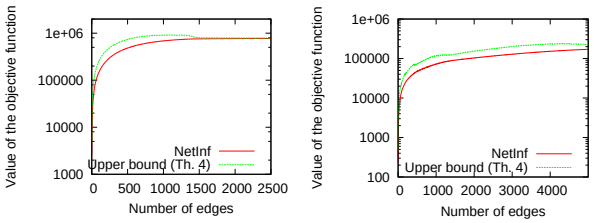
Algorithm 1 The NETINF Algorithm

Require: C, k
  $G \leftarrow \bar{K}$;
  for all $c \in C$ **do**
    $T_c \leftarrow dagtree(c)$;
  while $|G| < k$ **do**
    for all $(j, i) \in C \backslash G$ **do**
      $\delta_{j,i} = 0$, $M_{j,i} \leftarrow \emptyset$;
      for all $c : (j, i) \in c$ **do**
        let $w_c(m, n)$ be the weight of (m, n) in $G \cup \{(j, i)\}$;
        if $w_c(j, i) \geq w_c(Par_{T_c}(i), i)$ **then**
          $\delta_{j,i} = \delta_{j,i} + w_c(j, i) - w_c(Par_{T_c}(i), i)$;
          $M_{j,i} \leftarrow M_{j,i} \cup \{c\}$;
    $(j^*, i^*) \leftarrow \arg\max_{(j,i) \in C \backslash G} \delta_{j,i}$;
    $G \leftarrow G \cup \{(j^*, i^*)\}$;
    for all $c \in M_{j^*,i^*}$ **do**
      $Par_{T_c}(i^*) \leftarrow j^*$;
  return G;



(a) FF: Cascades per edge (b) FF: Cascade size

Figure 3: Number of cascades per edge and cascade sizes for a Forest Fire network ($1,000$ nodes, $1,477$ edges) with forward burning prob. 0.20, backward burning prob. 0.17 and exponential transmission model with parameters $\alpha = 1, \beta = 0.5$.



(a) Kronecker network (b) MemeTracker dataset

Figure 4: Score achieved by NETINF in comparison with the online upper bound from Theorem 4. In practice NETINF finds networks that are at 97% of NP-hard to compute optimal.

Speeding-up NETINF. We speed-up the algorithm by orders of magnitude by considering two improvements:

- *Localized update:* Consider the edge (j^*, i^*) selected by the *greedy algorithm* at iteration n and the trees $T_c : (j^*, i^*) \in T_c$. Now, fix an edge $(j, i) \notin G$ for which we know the marginal gain, $\delta_{j,i}$, at iteration n and we need to estimate it at iteration $n + 1$. The value of $\delta_{j,i}$ depends of cascades c for which $w_c(j, i) \geq w_c(Par_{T_c}(i), i)$. Then, if $i \neq i^*$, the value of $\delta_{j,i}$ at iteration n is the same as its value at iteration $n + 1$, otherwise, if $i = i^*$, we only need to update $\delta_{j,i}$ revisiting cascades c such that $(j^*, i^*) \in T_c$ and $w_c(j, i) \geq w_c(Par_{T_c}(i), i)$ just before selecting (j^*, i^*).
- *Lazy evaluation* can be used to drastically reduce the number of evaluations of marginal gains $F(G \cup \{e\}) - F(G)$ [21]. This procedure relies on the submodularity of F_C.

As we will show later, these two improvements decrease the run time by several order of magnitude without any loss in the solution quality. We call the algorithm that implements the greedy algorithm with the above speedups the NETINF algorithm (Algorithm 1). Additionally, NETINF lends itself to parallelization to tackle even bigger networks in shorter amounts of computation time.

4. EXPERIMENTAL EVALUATION

We first analyze the performance of NETINF on synthetic and real networks. We show that our algorithm outperforms a heuristic baseline and correctly discovers more than 90% of the edges.

4.1 Experiments on synthetic data

The goal of the experiments on synthetic data is to understand how the underlying network structure and the propagation model (exponential vs. power-law) affect the performance of our algorithm. In general, we proceed as follows: we generate a network G^*, simulate a set of cascades and for each cascade, record the node hit times t_u. Then, given the hit times, we try to recover the network G^*.

For example, Fig. 1(a) shows a graph G^* of 20 nodes and 23 edges. We generated 24 cascades and recovered G^*. A baseline method (b) (described below) performed poorly while our method (c) almost recovered G^* perfectly by making only two errors.

Experimental setup. We consider two models of directed real-world networks, namely, the Forest Fire model [20] and the Kronecker Graphs model [19] to generate G^*. For Kronecker graphs, we consider three sets of parameters that produce networks with a very different structure: a random graph [8] (Kronecker parameter matrix $[0.5, 0.5; 0.5, 0.5]$), a network with hierarchical community structure [5] ($[0.962, 0.107; 0.107, 0.962]$) and a core-periphery network [22] ($[0.962, 0.535; 0.535, 0.107]$). Notice that Forest Fire generates a scale free network [4].

We then simulate cascades on G^* using the generative model defined in Section 2.1 that is parameterized by α, which controls how quickly a cascade spreads, and β, that controls how far a cascade spreads. We pick cascade starting nodes uniformly at random and generate enough cascades so that 99% of the edges in G^* participate in at least one cascade.

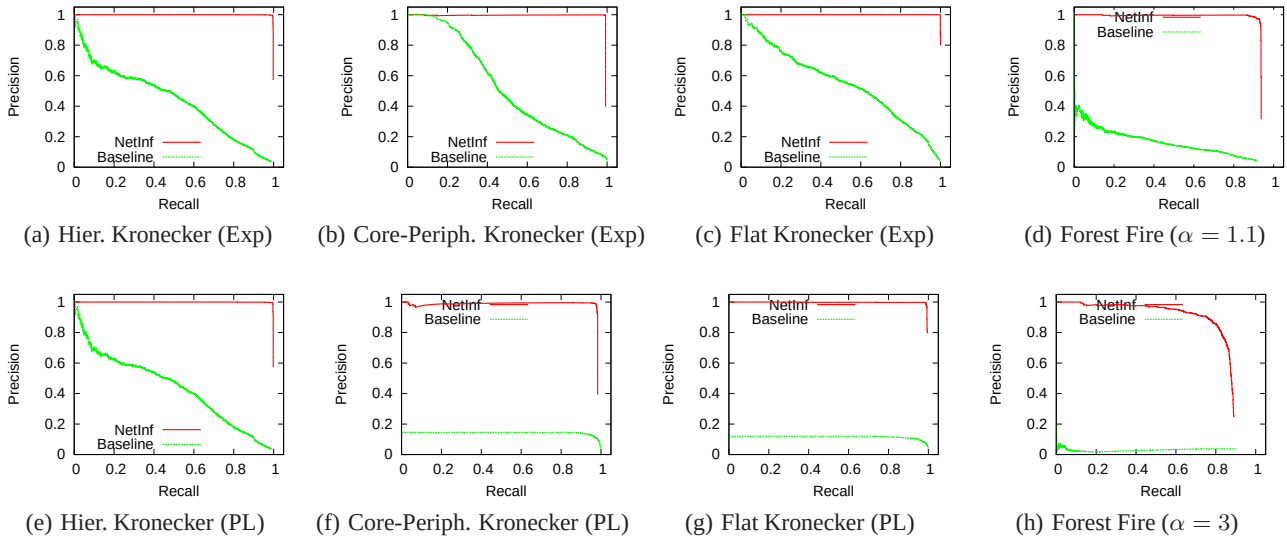
For example, for the Forest Fire network on 1,000 nodes and 1,477 edges, we generated 834 cascades. The majority of edges took part in 4 to 12 cascades and the cascade size distribution follows a power law (Figure 3(b)). The average and median number of cascades per edge are 9.1 and 8, respectively (Figure 3(a)).

Baseline method. To infer a diffusion network $\hat{G}$, we consider the following baseline method: For each *possible* edge (u, v), we compute $w(u, v) = \sum_{c \in C} P_c(u, v)$, i.e., overall how likely were the cascades $c \in C$ to propagate from u to v. Then we simply pick the k edges (u, v) with the highest weight $w(u, v)$ to obtain $\hat{G}$ (Fig. 1(b)).

Solution quality. We evaluate the performance of the NETINF algorithm in two different ways. First, we are interested in how successful NETINF is at optimizing the objective function that is NP-hard to optimize exactly. Using the online bound in Theorem 4, we can assess at most how far from the unknown optimal the NETINF solution is.

Figure 4(a) plots the value of the objective function as a function of the number of edges in $\hat{G}$. In red we plot the value achieved by NETINF and in green the upper bound using Thm. 4. This tells us that the value of the unknown optimal solution (that is NP-hard to compute exactly) is somewhere between the red and green curve. Notice that the band between optimal and the NETINF is relatively small. For example, at 2,000 edges in $\hat{G}$, NETINF finds the solution that is least 97% of optimal for synthetic data. Moreover, also notice a strong diminishing return effect, where the value of the objective function flattens out after about 1,000 edges. This means that, in practice, very sparse solutions (almost tree-like diffusion graphs) already achieve very high values of the objective function close to the optimal.

Accuracy of NETINF. We also evaluate our approach by studying how many edges inferred by NETINF are actually present in the true network G^*. We measure the precision and recall of our method. For every value of k ($1 \leq k \leq n^2$) we generate $\hat{G}_k$ on k edges by using NETINF or the baseline method. We then compute precision (what fraction of edges in $\hat{G}_k$ is also present G^*) and recall (what fraction of edges of G^* appears in $\hat{G}_k$). For small k, we expect low recall and high precision as we select the few edges that we are the



(a) Hier. Kronecker (Exp) (b) Core-Periph. Kronecker (Exp) (c) Flat Kronecker (Exp) (d) Forest Fire ($\alpha = 1.1$)

(e) Hier. Kronecker (PL) (f) Core-Periph. Kronecker (PL) (g) Flat Kronecker (PL) (h) Forest Fire ($\alpha = 3$)

Figure 5: (a-c, e-g): Precision and recall for three 1024 node Kronecker networks with exponential (Exp) and power law (PL) transmission model. (d,h): Precision and recall for a 1,000 node Forest Fire network with a power law transmission model. NETINF achieves break-even points of more than 0.9 regardless of the propagation model and underlying diffusion network structure.

most confident in. As k increases, precision will generally start to drop but the recall will monotonically increase.

Figure 5 shows the precision-recall curves of NETINF and the baseline method on three different Kronecker graphs (random, hierarchical community structure and core-periphery structure) with 1024 nodes and two cascade transmission models. The cascades were generated with an exponential transmission model with $\alpha = 1$, a power law transmission model with $\alpha = 2$ and a value of β low enough to avoid generating too large cascades (i.e. $\beta = 0.5$ for hierarchical, $\beta = 0.4$ for random and $\beta = 0.1$ for core-periphery). We generated between 2,000 and 4,000 cascades so that 99% of the edges participated in at least one cascade. We chose cascade starting points uniformly at random.

First, we focus on Figures 5(a), 5(b) and 5(c) where we use the exponential transmission model. Notice that the baseline method achieves the break-even point[1] in between 0.4 and 0.5 on all three networks. However, our method performs much better with the break-even point of 0.99 over all three datasets. This is a remarkable result as we were especially careful not to generate too many cascades since more cascades mean more evidence that makes the problem easier. Thus, using a very small number of cascades, where every edge of G^* participates in only a few cascades, we can almost perfectly recover the underlying diffusion network G^*.

Similarly, Figures 5(e), 5(f) and 5(g) show the performance on the same three networks but using the power law transmission model. The performance of the baseline now dramatically drops. This is due to the fact that the variance of power-law (and heavy tailed distributions in general) is much larger than the variance of an exponential distribution. Thus the diffusion network inference problem is much harder in this case. As the baseline pays high price due to the increase in variance with the break-even point dropping below 0.1 the performance of NETINF remains stable.

We also examine the results on the Forest Fire network (Figures 5(d) and 5(h)). Again, the performance of the baseline is very low while NETINF achieves the break-even point at around 0.90.

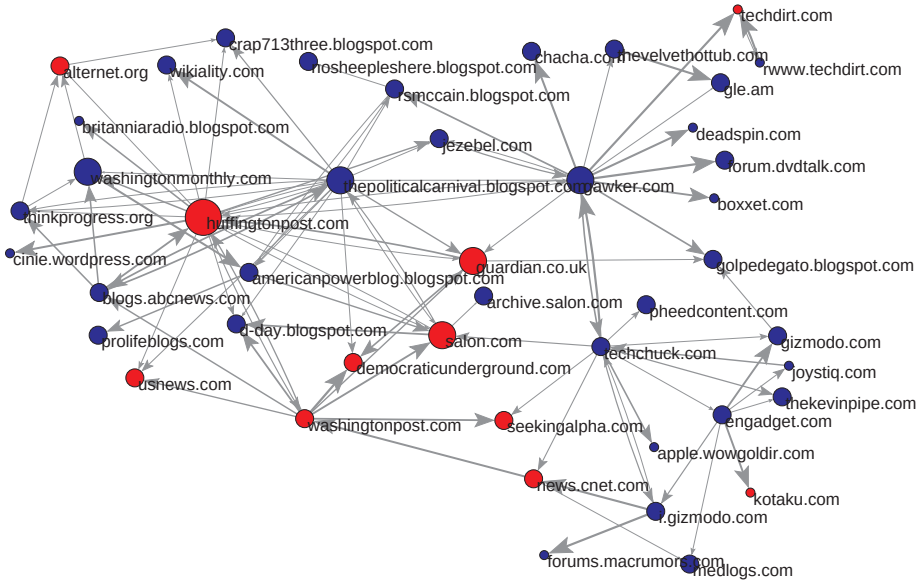
[1]The point at which recall is equal to precision.

Generally, the performance on the Forest Fire network is a bit lower than on the Kronecker graphs. However, it is important to note that while these networks have very different global network structure (from hierarchical, random, scale free to core periphery) the performance of NETINF is remarkably stable and does not seem to depend on the structure of the network we are trying to infer or the particular type of cascade transmission model.

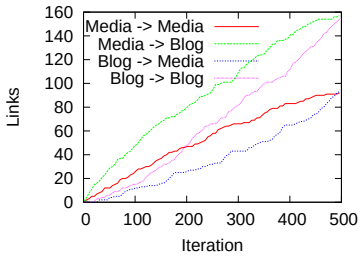
Performance vs. cascade coverage. Intuitively, the larger the number of cascades that spread over a particular edge the easier it is to identify it. In our experiments so far, we carefully generated a relatively small number of cascades. Next, we examine how the performance of NETINF depends on the amount of available cascade data. Fig. 7(b) plots the performance of NETINF (break-even point) as a function of the available cascade data measured in the number of transmissions over all cascades, i.e., $x = 1$ means that the total number of transmission events (sum of cascade sizes) used for the experiment was equal to the number of edges in G^*. Small values of β produce larger cascades, increasing the difficulty of our problem. Note that NETINF requires a total number of transmission events between 2 and 5 times the number of edges in G^* to successfully recover most of the edges of G^*.

Stopping criterion. In practice one does not know how long to run the algorithm and how many edges to insert into the network $\hat{G}$. Given the results from Figure 4, we found the following heuristic to give good results. We run the NETINF algorithm for k steps where k is chosen such that the objective function is "close" to the upper bound, i.e., $F_C(\hat{G}) > x \cdot \text{OPT}$, where OPT is obtained using the online bound. In practice we use values of x in range 0.8–0.9.

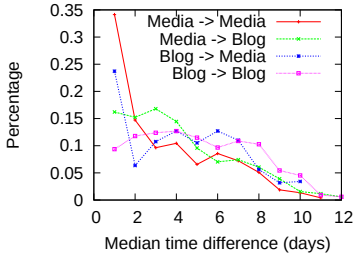
Scalability. Figure 7(a) shows the average computation time per edge added of our algorithm implemented with lazy evaluation and localized update. We use a hierarchical Kronecker network and an exponential transmission model with $\alpha = 1$ and $\beta = 0.5$. Localized update speeds up the algorithm for an order of magnitude (45×) and lazy evaluation further gives a factor of 6 improvement. Thus, overall, we achieve two orders of magnitude speed up (280×), without *any* loss in solution quality.



(a) Small part of news media (red) and blog (blue) diffusion network



(b) Edges by time added



(c) Median edge time lag

Figure 6: (a) A small part of the larger diffusion network. (b) Number and strength of edges between different media types. Edges of news media influencing blogs are the strongest. (c) Median time lag on edges of different type.

4.2 Experiments on real data

Dataset description. We use more than 172 million news articles and blog posts from 1 million online sources over a period of one year from September 1 2008 till August 31 2009[2]. Based on this raw data, we use two different methodologies to trace information on the Web. First, we use hyperlinks between blog posts to trace the flow of information [23]. As the use of hyperlinks to refer to the source of information is relatively rare (especially in mainstream media), we also use the MemeTracker [18] methodology to extract more than 343 million short textual phrases (like, "Joe, the plumber" or "lipstick on a pig"). Out of these, 8 million distinct phrases appeared more than 10 times, with the cumulative number of mentions of over 150 million. We cluster the phrases to aggregate different textual variants of the same phrase [18]. We then consider each phrase cluster as a separate cascade c. Since all documents are time stamped, a cascade c is simply a set of timestamps when websites first mentioned a phrase in the phrase cluster c. So, we observe the times when sites mention a particular phrase but not where they copied that phrase from. For the experiments here we use the top 1,000 media sites and blogs with the largest number of documents. We then consider the largest 5,000 cascades (phrase clusters) and for each website we record the time when they first mention a phrase in the particular phrase cluster. Note that cascades in general do not spread over all the sites, which our methodology can successfully handle.

Visualization of diffusion networks. First we examine the structure of the inferred network. Figure 6(a) shows the largest connected component of the diffusion network after 100 edges have been chosen. The size of the nodes is proportional to the number of articles on the site and the width of the edge is proportional to the probability of influence, i.e., stronger edges have higher width. Here we used the hyperlinks to trace the spread of information. Since news media articles rarely use hyperlinks to refer to one another, the network is somewhat biased towards web blogs (blue nodes). There are several interesting patterns. First, notice how three main clusters emerge: on the left side of the network we can see blogs and news media sites related to politics, at the right top, we have blogs devoted to gossip, celebrity news or entertainment and on the right bottom, we can distinguish blogs and news media sites that deal with technological news. As Huffington Post and Political Carnival play the central role on the political side of the network, mainstream media sites like Washington Post, Guardian and professional blog Salon.com play the role of connectors between the different parts of the network. The celebrity gossip part of the network is dominated by the blog Gawker and technology news gather around blogs Gizmodo and Engadget, with CNet and TechChuck establishing the connection to the rest of the network. For reasons of space, we refer the reader to the supporting website [1] for additional graphs.

Insights into the diffusion on the web. The inferred diffusion networks also allow for analysis of the global structure of information propagation on the Web. For this analysis, we use the MemeTracker phrase clusters as cascades and analyze the structure of the inferred information diffusion network.

Figure 6(b) investigates the number of links in the inferred network that point between different types of sites. Here we split the sites into mainstream media and blogs. Notice that most of the links point from news media to blogs, which says that most of the time information propagates from the mainstream media to blogs. Then notice how at first many media-to-media links are chosen but in later iterations the increase of these links starts to slow down. This means that media-to-media links tend to be the strongest and NETINF picks them early. The opposite seems to occur in case of blog-to-blog links where relatively few are chosen first but later the algorithm picks more of them. Lastly, links capturing the influence of blogs on mainstream media are the rarest and weakest. This suggests that most information travels from mass media to blogs.

[2]Data available at `http://memetracker.org` and `http://snap.stanford.edu/netinf`

In Figure 6(c), we investigate the median time difference between mentions of different types of sites. For every edge of the inferred diffusion network, we compute the median time needed for the information to spread from the source to the destination node. Again, we distinguish the mainstream media sites and blogs. Notice that media sites are quick to infect one another or even to get infected from blogs. However, blogs tend to be much slower in propagating information. It takes a relatively long time for them to get "infected" with information regardless whether the information comes from the mainstream media or the blogosphere.

Solution quality. Similarly as with synthetic data, in Figure 4(b) we also investigate the value of the objective function and compare it to the online bound. Notice that the bound is almost as tight as in the case of synthetic networks, finding the solution that is least 84% of optimal and both curves are similar in shape to the synthetic case value. Again, as in the synthetic case, the value of the objective function quickly flattens out which means that one needs a relatively few number of edges to capture most of the information flow on the Web.

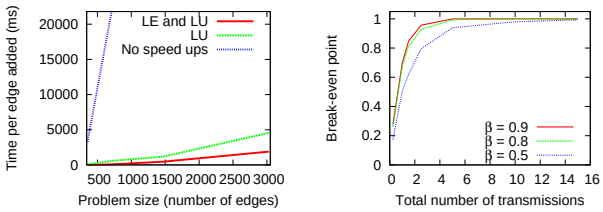
Accuracy on real data. As there is not objective ground truth network on real data, we perform the following experiment. We create a network where there is an edge (u, v) if a webpage on a site u linked to a page on a site v. we take the top 500 media sites and blogs in terms of number of hyperlinks and the top 4,000 hyperlink in terms of number of posts links. First, we consider this as the ground truth network G^*. We use the hyperlink cascades to infer the network $\hat{G}$ and evaluate how many edges NETINF got right. Figure 8(a) shows the performance of NETINF and the baseline. Notice that the baseline method achieves the break-even point of 0.34, while our method performs better with a break-even point of 0.44. Second, we try a considerably harder problem, we use the cascades based on the MemeTracker phrase clusters to infer G^*. Figure 8(b) shows the performance of NETINF and the baseline. The baseline method has a break-even point of 0.17 and NETINF achieves a break-even point of 0.28. To have a fair comparison with the synthetic cases, notice that the exponential transmission model is a simplistic assumption for our real dataset, and NETINF can get additional mileage with respect to the baseline choosing a more complex transmission model.

5. RELATED WORK

There are several lines of work we build upon. Although the information diffusion in on-line settings has received considerable attention [2, 11, 16, 17, 23, 24, 25], only a few studies were able to study the actual shapes of cascades [23, 25]. The problem of inferring links of diffusion was first studied by Adar and Adamic [2], who formulated it as a supervised classification problem and used Support Vector Machines combined with rich textual features to predict the occurrence of individual links. Although rich textual features are used, links are predicted independently and thus their approach is similar to our baseline method in the sense that it picks a threshold (i.e., hyperplane in case of SVMs) and predicts individually the most probable links.

Network structure learning has been considered for estimating the dependency structure of probabilistic graphical models [9, 10] and for estimating epidemiological networks [29]. In both cases, the problem is formulated in a probabilistic framework. However, since the problem is intractable, heuristic greedy hill-climbing or stochastic search that offer no performance guarantee were usually used in practice. In contrast, our work provides a novel formulation and a *tractable* solution together with an approximation guarantee.

Last, although *submodular* maximization has been previously considered for sensor placement [21] and finding influencers in viral marketing [13], to the best of our knowledge, the present work is the first that considers submodular function maximization in the context of network structure learning.



(a) Scalability (b) Amount of cascade data

Figure 7: (a): Average time per edge added by our algorithm implemented with lazy evaluation (LE) and localized update (LU). (b): Performance of NETINF as a function of the amount of cascade data. On average NETINF requires about two propagation events per edge of the original network in order to reliably recover the true network structure.

6. CONCLUSIONS

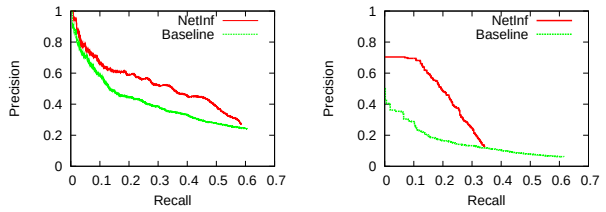
We have investigated the problem of tracing paths of diffusion and influence. We formalized the problem and developed a scalable algorithm, NETINF, to infer networks of influence and diffusion. First, we defined a generative model of cascades and showed that choosing the best set of k edges maximizing the likelihood of the data is NP-hard. By exploiting the submodularity of our objective function, we developed NETINF, an efficient algorithm for inferring a near-optimal set of k directed edges. By exploiting localized updates and lazy evaluation, our algorithm is able to scale to very large real data sets.

We evaluated our algorithm on synthetic cascades sampled from our generative model, and showed that NETINF is able to accurately recover the underlying network from a relatively small number of samples. In our experiments, NETINF drastically outperformed a naive maximum weight baseline heuristic.

Most importantly, our algorithm allows us to study properties of real networks. We evaluated NETINF on a large real data set of memes propagating across news websites and blogs. We found that the inferred network exhibits a core-periphery structure with mass media influencing most of the blogosphere. Clusters of sites related to similar topics emerge (politics, gossip, technology, etc.), and a few sites with social capital interconnect these clusters allowing a potential diffusion of information among sites in different clusters.

There are several interesting directions for future work. Here we only used time difference to infer edges and thus it would be interesting to utilize more informative features (e.g., textual content of postings etc.) to more accurately estimate the influence probabilities. Moreover, our work considers static propagation networks, however real influence networks are dynamic and thus it would be interesting to relax this assumption. Last, there are many other domains where our methodology could be useful: inferring interaction networks in systems biology (protein-protein and gene interaction networks), neuroscience (inferring physical connections between neurons) and epidemiology.

We believe that our results provide a promising step towards understanding complex processes on networks based on partial observations.



(a) Hyperlinks cascades (b) Memetracker cascades

Figure 8: Precision and recall for a 500 node hyperlink network using (a) hyperlinks cascades and (b) MemeTracker cascades.

Acknowledgments

We thank Spinn3r for resources that facilitated the research. Manuel Gomez Rodriguez has been supported in part by a Fundacion Caja Madrid Graduate Fellowship. The research was supported in part by generous gifts by Microsoft, Yahoo and IBM, and under the auspices of the DOE by LLNL under contract DE-AC52-07NA27344, ONR N000140911044 and NSF CNS0932392 and IIS0953413.

7. REFERENCES

[1] Supporting website. http://snap.stanford.edu/netinf.

[2] E. Adar and L. A. Adamic. Tracking information epidemics in blogspace. In *Web Intelligence*, pages 207–214, 2005.

[3] A.-L. Barabási. The origin of bursts and heavy tails in human dynamics. *Nature*, 435:207, 2005.

[4] A.-L. Barabási and R. Albert. Emergence of scaling in random networks. *Science*, 286:509–512, 1999.

[5] A. Clauset, C. Moore, and M. E. J. Newman. Hierarchical structure and the prediction of missing links in networks. *Nature*, 453(7191):98–101, 2008.

[6] R. Crane and D. Sornette. Robust dynamic classes revealed by measuring the response function of a social system. *Proc. of the National Academy of Sciences*, 105(41):15649–15653, October 2008.

[7] J. Edmonds. Optimum branchings. *Journal of Resesearch of the National Bureau of Standards*, (71B):233–240, 1967.

[8] P. Erdős and A. Rényi. On the evolution of random graphs. *Publication of the Mathematical Institute of the Hungarian Academy of Science*, 5:17–67, 1960.

[9] N. Friedman and D. Koller. Being Bayesian about network structure. A Bayesian approach to structure discovery in Bayesian networks. *Machine Learning*, 50(1):95–125, 2003.

[10] N. Friedman, I. Nachman, and D. Pe'er. Learning Bayesian network structure from massive datasets: The "Sparse Candidate" algorithm. In *Proc. UAI*, 1999.

[11] D. Gruhl, R. Guha, D. Liben-Nowell, and A. Tomkins. Information diffusion through blogspace. In *WWW '04: Proc. of the 13th international conference on World Wide Web*, pages 491–501, 2004.

[12] E. Katz and P. Lazarsfeld. *Personal influence: The part played by people in the flow of mass communications*. Free Press, 1955.

[13] D. Kempe, J. M. Kleinberg, and E. Tardos. Maximizing the spread of influence through a social network. In *KDD '03: Proc. of the 9th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 137–146, 2003.

[14] S. Khuller, A. Moss, and J. Naor. The budgeted maximum coverage problem. *Information Processing Letters*, 70(1):39–45, 1999.

[15] D. Knuth. *The art of computer programming*. Addison-Wesley, 1968.

[16] R. Kumar, J. Novak, P. Raghavan, and A. Tomkins. Structure and evolution of blogspace. *CACM*, 47(12):35–39, 2004.

[17] J. Leskovec, L. A. Adamic, and B. A. Huberman. The dynamics of viral marketing. In *EC '06: Proc. of the 7th ACM conference on Electronic commerce*, pages 228–237, 2006.

[18] J. Leskovec, L. Backstrom, and J. Kleinberg. Meme-tracking and the dynamics of the news cycle. In *KDD '09: Proc. of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 497–506, New York, NY, USA, 2009. ACM.

[19] J. Leskovec and C. Faloutsos. Scalable modeling of real graphs using kronecker multiplication. In *ICML '07: Proc. of the 24th International Conference on Machine Learning*, page 504, 2007.

[20] J. Leskovec, J. Kleinberg, and C. Faloutsos. Graphs over time: densification laws, shrinking diameters and possible explanations. In *KDD '05: Proc. of the 11th ACM SIGKDD international conference on Knowledge discovery in data mining*, page 187, 2005.

[21] J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. VanBriesen, and N. Glance. Cost-effective outbreak detection in networks. In *KDD '07: Proc. of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 420–429, 2007.

[22] J. Leskovec, K. J. Lang, A. Dasgupta, and M. W. Mahoney. Statistical properties of community structure in large social and information networks. In *WWW '08: Proc. of the 17th International Conference on World Wide Web*, 2008.

[23] J. Leskovec, M. McGlohon, C. Faloutsos, N. Glance, and M. Hurst. Cascading behavior in large blog graphs. In *SDM '07: Proc. of the SIAM Conference on Data Mining*, 2007.

[24] J. Leskovec, A. Singh, and J. M. Kleinberg. Patterns of influence in a recommendation network. In *PAKDD '06: Proc. of the 10th Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 380–389, 2006.

[25] D. Liben-Nowell and J. Kleinberg. Tracing the flow of information on a global scale using Internet chain-letter data. *Proc. of the National Academy of Sciences*, 105(12):4633–4638, 25 Mar. 2008.

[26] R. D. Malmgren, D. B. Stouffer, A. E. Motter, and L. A. A. N. Amaral. A poissonian explanation for heavy tails in e-mail communication. *Proc. of the National Academy of Sciences*, 105(47):18153–18158, November 2008.

[27] G. Nemhauser, L. Wolsey, and M. Fisher. An analysis of approximations for maximizing submodular set functions. *Mathematical Programming*, 14(1):265–294, 1978.

[28] E. M. Rogers. *Diffusion of Innovations*. Free Press, New York, fourth edition, 1995.

[29] J. Wallinga and P. Teunis. Different epidemic curves for severe acute respiratory syndrome reveal similar impacts of control measures. *American Journal of Epidemiology*, 160(6):509–516, 2004.

[30] D. J. Watts and P. S. Dodds. Influentials, networks, and public opinion formation. *Journal of Consumer Research*, 34(4):441–458, December 2007.