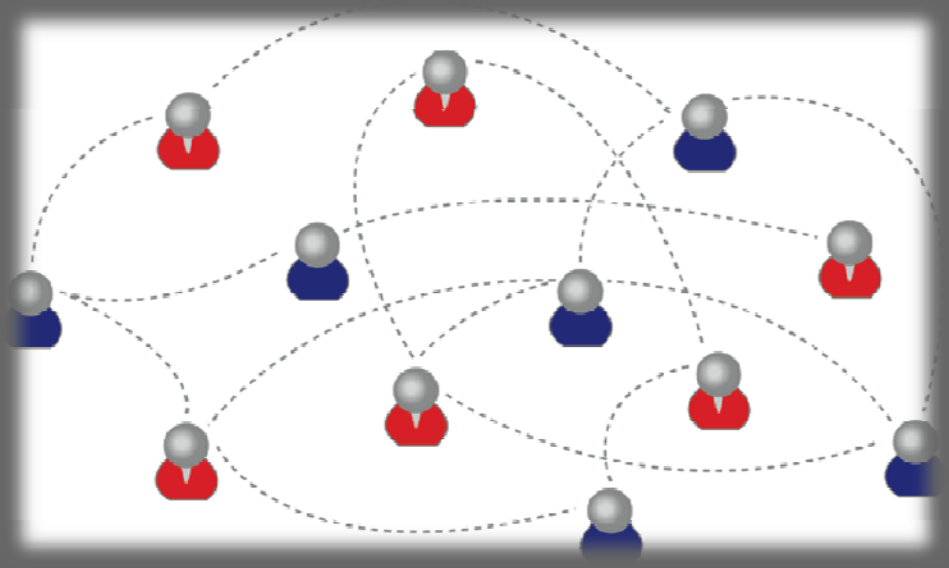


INFERRING NETWORKS OF DIFFUSION AND INFLUENCE

Presented by Alicia Frame

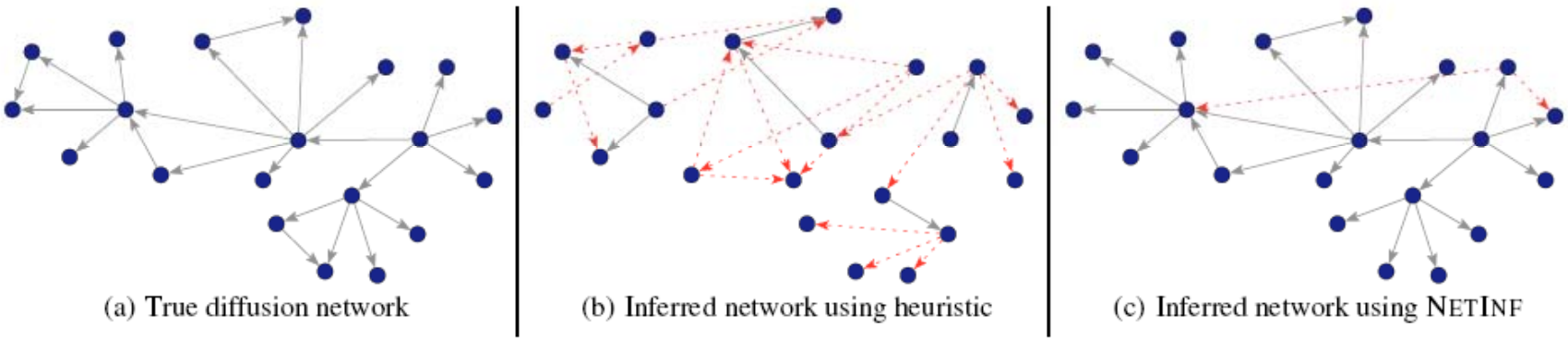
Paper by Manuel Gomez-Rodriguez, Jure Leskovec, and Andreas Kraus

Introduction

- Network diffusion is an important process – information spread, epidemiology
- Challenges:
  - To track cascading processes, you need to identify the contagion and how to trace it
  - Diffusion takes place on a network but this network is usually unknown and unidentified
  - Know **when** a node is infected, but not by **whom**

Introduction

- Questions:
  1. What is the network over which information propagates
  2. What is the global structure of the network?
  3. How do news media and blogs interact



(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.

Problem Formulation

- Assumptions:
  - Many different cascades propagate over an unknown static network
  - Observe when nodes get infected, but not by whom
- Goal:
  - Infer the unknown network over which cascades propagate
  - Infer the network where a directed edge (u,v) means that node v tends to be infected after node u

Example

- Network is made up of news sites and blogs on the web
- Each cascade is a different piece of information spreading through the network
- Know when a piece of information was mentioned on a site
- And edge (u,v) means that a site v tends to repeat stories after a site u

Problem Statement

- Given a hidden network G^*, observe multiple cascades to get an estimated version of the network, $\hat{G}$
- Each cascade leaves a trace $(u_i, t_i, \varphi_i)_c$
  - Cascade c reached node u_i at time t_i with a set of attributes φ_i
  - If a node is not hit by a cascade then $t_u = \infty$
- A cascade is fully specified by
  - Vector $\mathbf{t} = [t_1, \ldots, t_n]$ of hit times
  - Feature vector $\boldsymbol{\phi} = [\phi_1, \ldots, \phi_n]$ describing the properties of the contagion and the node

Model Formulation

- Assumptions:
  - For a fixed cascade $c=(\mathbf{t}, \boldsymbol{\phi})$, we know which nodes influenced other nodes
  - Every node *v* in a cascade is influence by at most one node *u*
  - Each cascade is given by a directed tree, T, which is contained in G
- Probabilistic model:
  - Cascade Transmission Model
  - Cascade Propagation Model
  - Network Inference Model
- NetInf algorithm

Cascade Transmission Model

- How likely is it the a node *u* spreads the cascade c to a node *v*
  - A node infects each of its neighbors independently
  - Ignore multiple infections because the first is sufficient
- $P_c(u,v)$ is the conditional probability of observing cascade c spreading
  - Cascades only propagate forward in time → if $t_u \geq t_v$, $P_c(u,v)=0$
  - Probability of transmission depends only on the time difference between node hit times:

$$\Delta = t_v - t_u$$

Cascade Transmission Model

- Need to determine the time, t_v, when *u* spreads the cascade to *v*
  - Probability (1-β) that the cascade stops before v and $t_v = \infty$
  - Otherwise, $t_v = t_u + \Delta$
  - Consider power law and exponential models of waiting time
- Given the probability $P_c(u,v)$, you can define the probability of observing cascade c propagating in a particular tree structure T

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

Cascade Propagation Model

- We know the probability of a single cascade c propagating in a particular tree T – P(c|T)
- Need to compute P(c|G), the probability that a cascade c occurs in a graph G
  - Combine the probabilities of individual trees into a probability of a cascade c occurring over a graph G
  - Consider all the ways c could have spread of G

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

- Define the probability of a set of cascades, C, occurring in C

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

Network Inference Problem

- Aim is to find the most likely graph, $\hat{G}$, that describes the observed cascades

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

- Computing the probability of each cascade, and then the probability of each tree, is intractable
  - Super exponential in the size of G
  - Can be improved to $O(|C|n^3)$, but that is still too expensive
  - Above formulation only evaluates the quality of a particular graph G, whereas we want the *best* graph

Proposed Algorithm

- Instead of considering every possible tree T, only consider the most likely propagation tree, T

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

- Define the improved of a cascade c under a graph G over an empty graph:

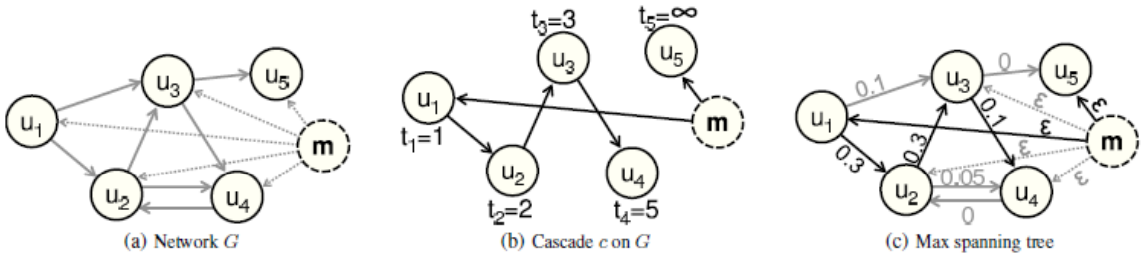
$$F_c(G) = \max_{T \in \mathcal{T}(G)} \log P(c|T) - \max_{T \in \mathcal{T}(\bar{K})} \log P(c|T)$$

- The maximum of $P(C|G) = F_C(G)$

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

Proposed Algorithm

- Introduce an additional node *m*, an external source that can infect any node *u*
  - Connect *m* to all nodes in the graph with an ε edge
- Most likely tree T is a *maximum weighted spanning tree* in G
  - Each edge (i,j) has weight $w_c(i,j)$ and $F_c(G)$ is the sum of the weighted edges in T



(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.

Proposed Algorithm

- Start with an empty graph, K
  - F_C is non negative and monotonic
  - Adding more edges does not degrease solution quality
  - The complete graph will maximize F_C
- We are interested inferring sparse graphs which only include a small number *k* of relevant edges

$$G^* = \underset{|G| \leq k}{\operatorname{argmax}} F_C(G)$$

- Solving this is NP hard

Proposed Algorithm

- You can prove that F_C is submodular
  - diminishing returns property
  - Allows you to find a near optimal solution to the problem
- Greedy algorithm
  - Start with empty graph $\bar{K}$
  - Iteratively add the edge e_i which maximizes marginal gain $$e_i = \underset{e \in G \setminus G_{i-1}}{\operatorname{argmax}} \; F_C(G_{i-1} \cup \{e\}) - F_C(G_{i-1})$$
  - Stop once it has slected k edges and return the solution

Proposed Algorithm

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;

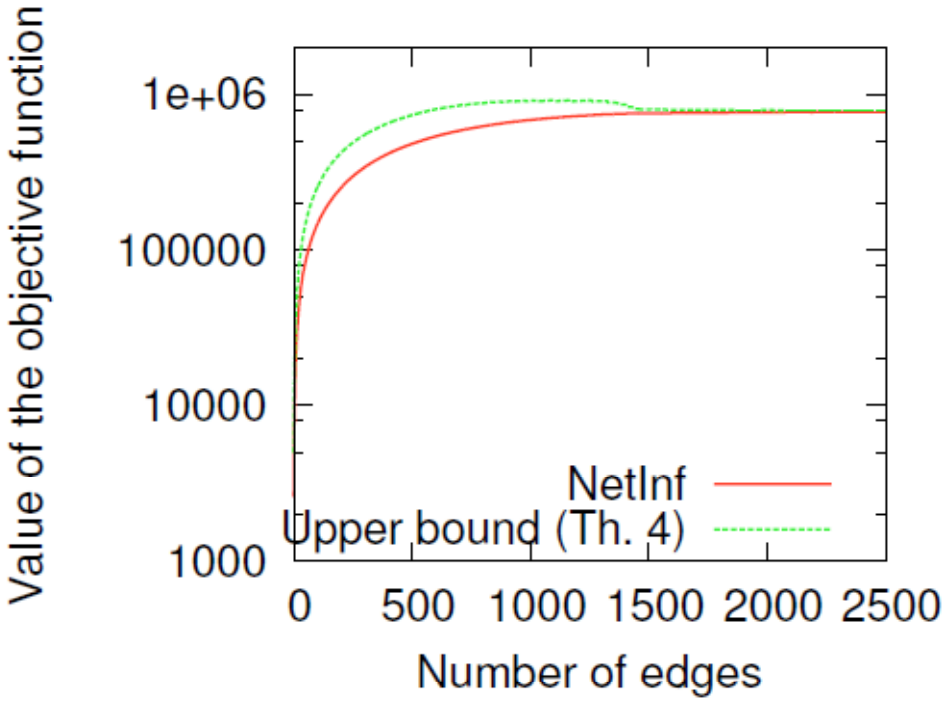
- Can be sped up with localized updates and lazy evaluations

Evaluation with Synthetic Data

- Forest fire model: essentially a scale free graph
- Kronecker Graph:
  - Random graph
  - Hierarchical community structure
  - Core periphery network
- Simulate cascades parameterized by how quickly the cascade spreads and how far it spreads, picking starting nodes at random

Experiments on Synthetic Data

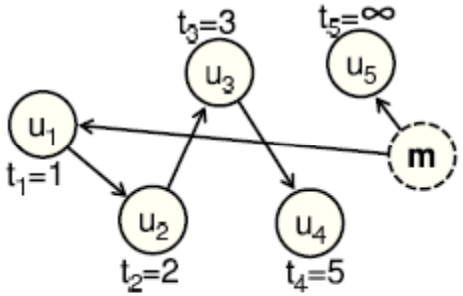
- Solution quality: how close does the NetInf algorithm get to the optimal solution



(a) Kronecker network

Experiments on Synthetic Data

- Accuracy: how many edges inferred by NetInf are present in the true network G^*
  - Precision: fraction of edges in G_k also in G^*
  - Recall: fraction of edges in G^* also in G_k
- Compared to 'baseline method'
  - For each *possible edge* (u,v) compute how likely were the cascades $c \in C$ to propagate from u to v
  - Pick the k edges with the highest weight



(b) Cascade c on G

Experiments on Synthetic Data

- NetInf performs better than the baseline in 97% of cases

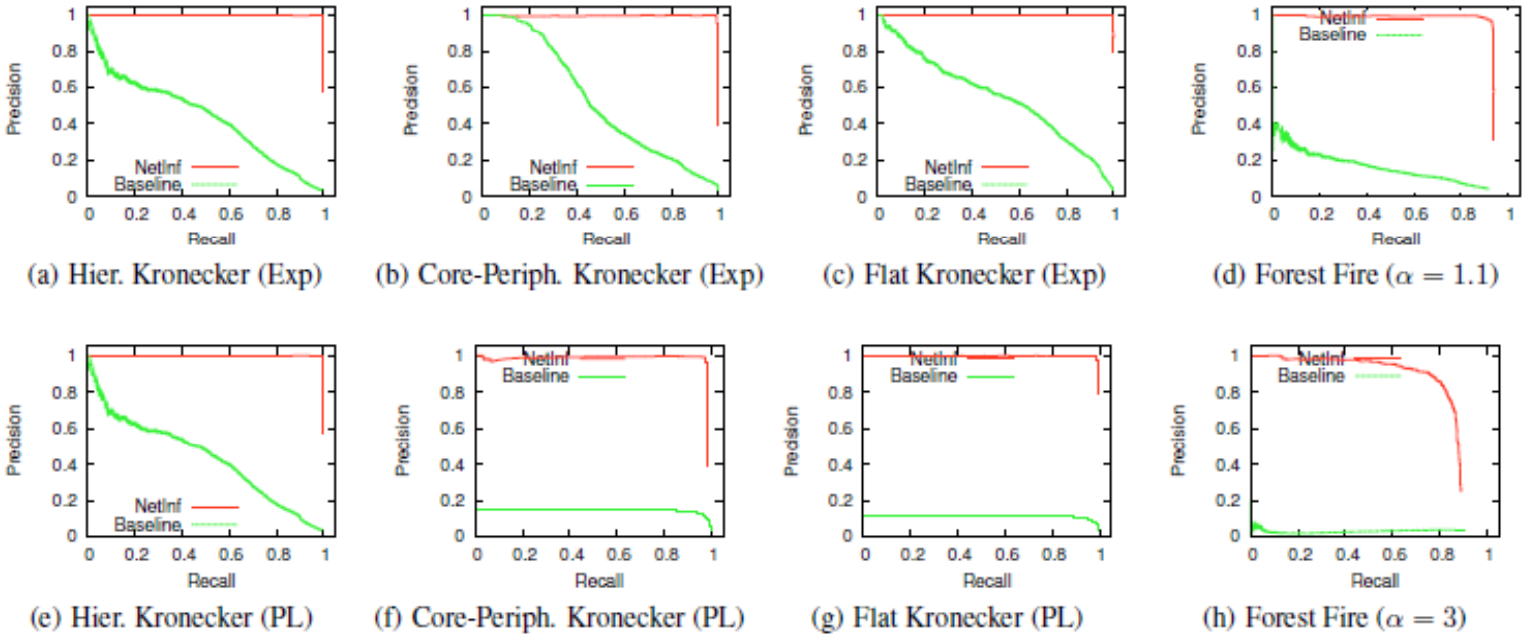
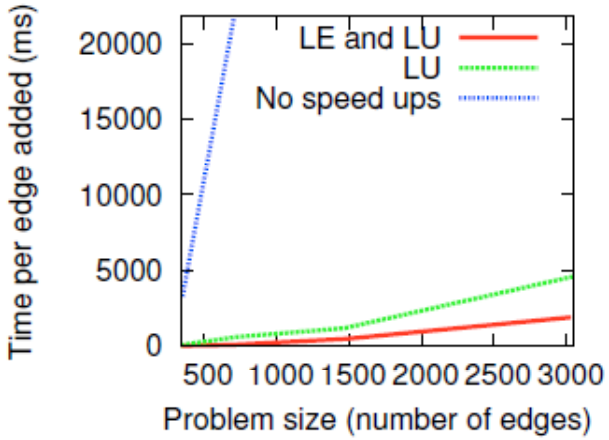


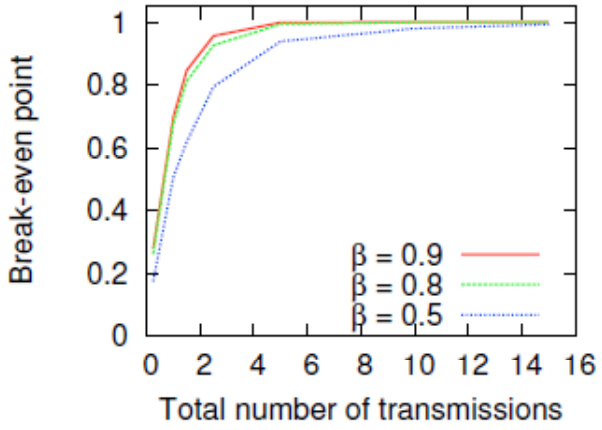
Figure 5: (a-c, e-g): Precision and recall for three 1024 node Kronecker networks with exponential (Exp) and power law (PL) transmission model. (d,h): Precision and recall for a 1,000 node Forest Fire network with a power law transmission model. NETINF achieves break-even points of more than 0.9 regardless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

Experiments on Synthetic Data:

- NetInf requires the total number of transmission events between 2 and 5 times the number of edges in G^*
- With lazy evaluation and localized update, computation time is two orders of magnitude faster



(a) Scalability

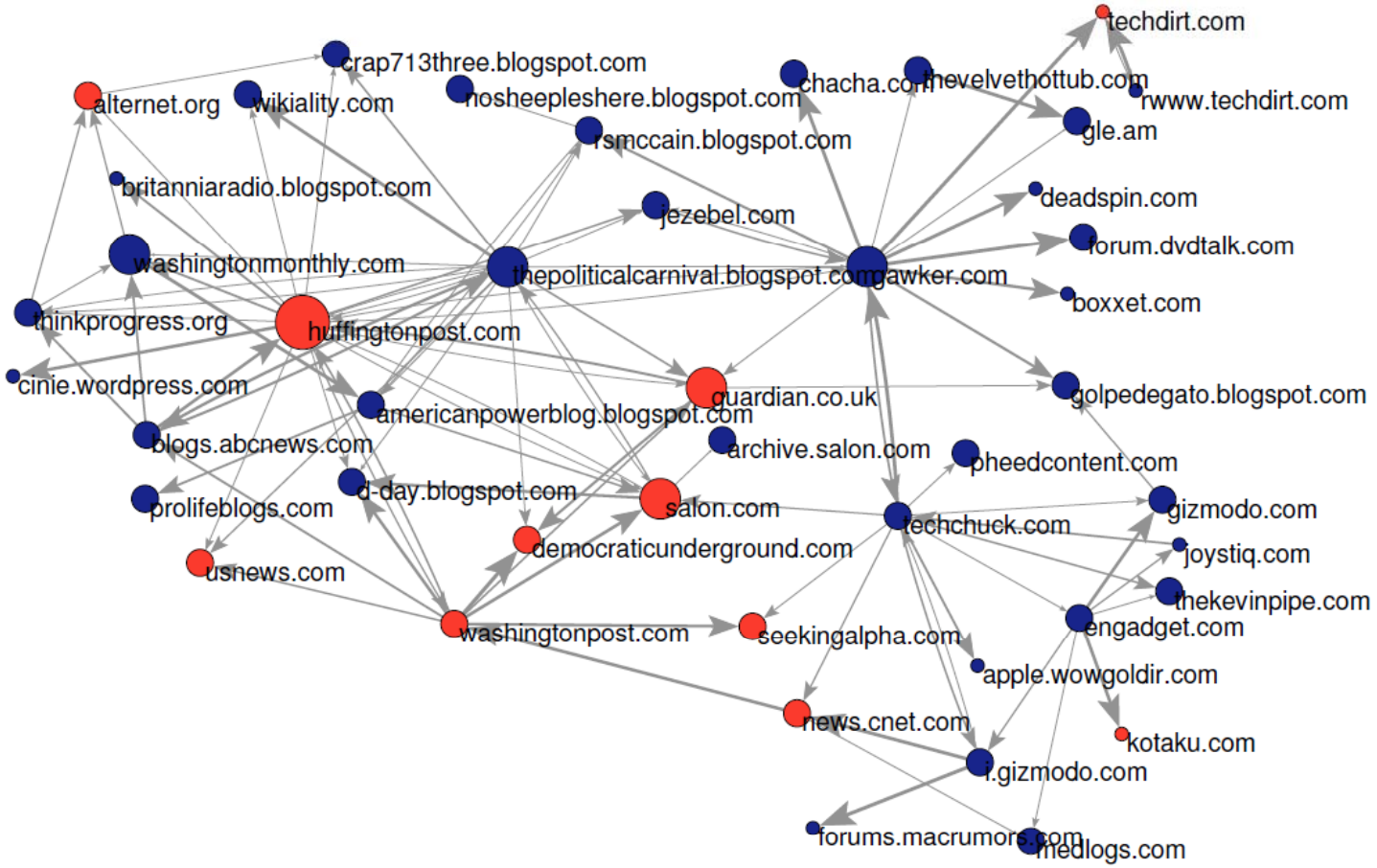


(b) Amount of cascade data

Experiments on Real Data

- Over 172 million news articles and blog posts
  - Used hyperlinks between blog posts to retrieve information
  - Also used 'memetracker' methodology
    - extracts short textual phrases
    - Cluster baased on different textual variants of the same phrase
    - Cascade is the set of time stamps
- Considered the top 1,000 media sites with the most documents and the 5,000 largest cascades

Experiments on Real Data



Largest connected component after 100 edges added

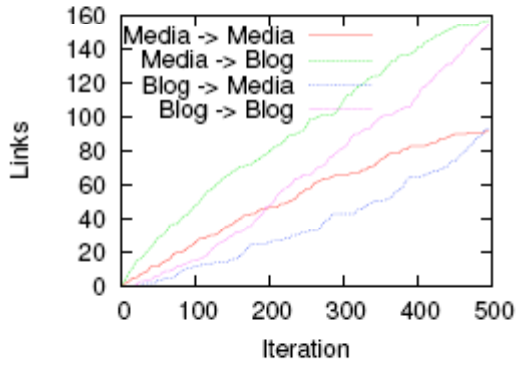
Using hyperlinks only

Experiments of Real Data

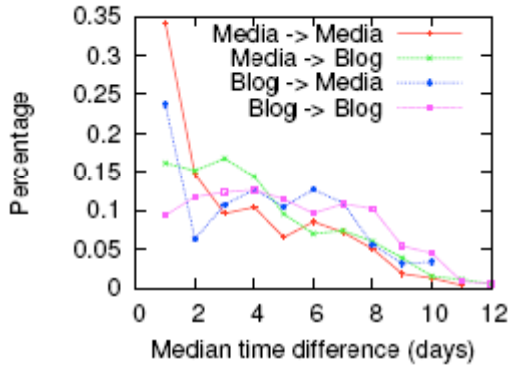
- Interesting patterns:
  - Clusters of sites related to politics, gossip, and technology
  - Mainstream media sites act as connectors between parts of the network
- Issues
  - Gawker media owns several of the prominent blogs, which all link to eachother
  - Typos in the nodes result in them showing up multiple times
  - Obscure blogs marked as ‘central’

Experiments on Real Data

- Also used memetracker to look at global structure of information propagation
  - Most information propagates from mainstream media to blogs
  - Media to media links are the strongest
  - Links capturing influence of blogs onto media are rare



(b) Edges by time added



(c) Median edge time lag

Conclusions

- Novel *tractable* solution to information propagation on networks with an approximation guarantee
  - Developed a generative model of information cascades
  - Exploiting the submodularity of the objective function, they developed NetInf to infer a near-optimal set of k directed edges
- Using synthetic data, found NetInf can accurately recover the underlying network
- Allows study of properties of real world networks

Discussion?

- Only applicable to static networks
- Requires full knowledge of ‘infection times’
- Requires many cascades to accurately infer graph
- Probably not extensible to their other examples
  - Epidemiology
  - There are already effective techniques for systems biology
- External node assumption?