Inferring Networks of Diffusion and Influence

Jure Leskovec

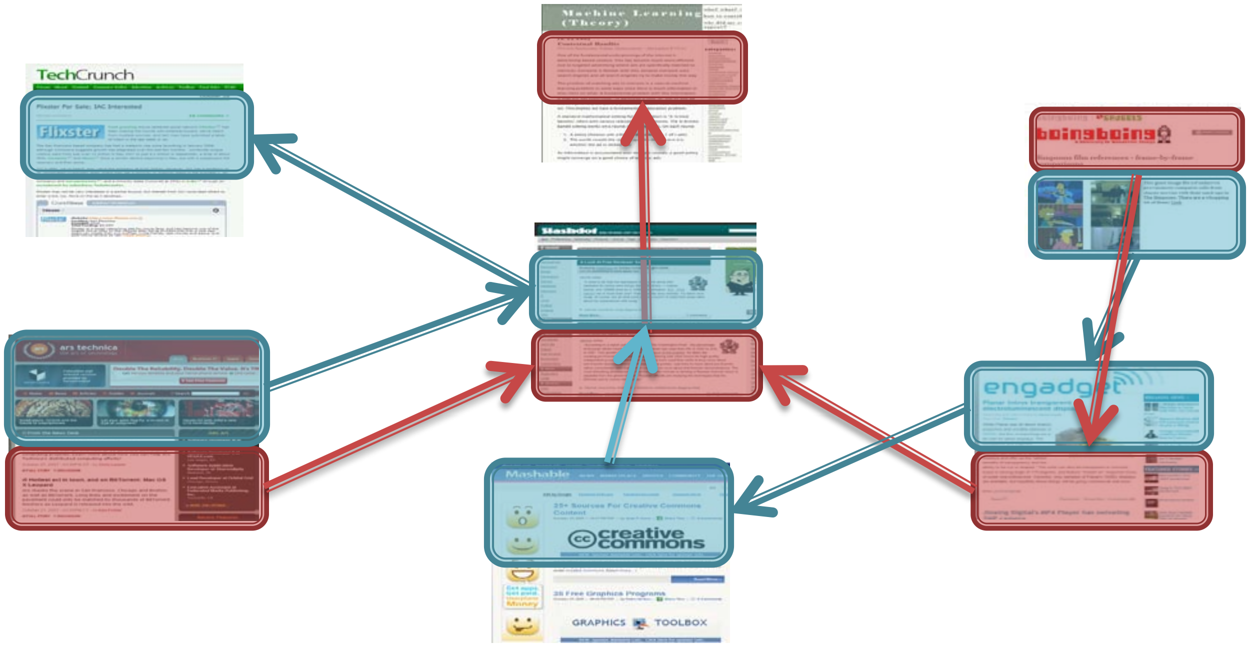
Joint work with Manuel Gomez-Rodriguez, and Andreas Krause

Networks and Processes on Them

- Many times it is hard to directly observe the underlying social network
  - Hidden/hard-to-reach populations:
    - Drug injection users
  - Implicit connections:
    - Network of information sharing in online media
- But it is often easier to observe results of the processes taking place on such (invisible) networks:
  - Virus propagation:
    - People get sick, they see the doctor
  - Information networks:
    - Blogs mention information

Information Diffusion Network

- Information diffuses through the network



- We only see the mention but not the source
- Can we reconstruct (hidden) **diffusion network**?

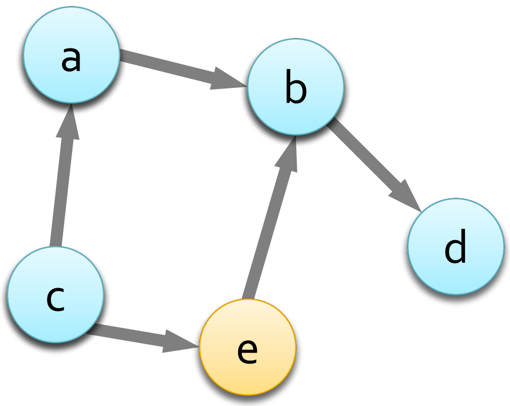
More examples

- Virus propagation:
  - Viruses propagate through the network
  - We only observe times when people get sick
    - But NOT who infected them
- Word of mouth & Viral marketing:
  - Recommendations and influence propagate
  - We only observe when people buy products
    - But Not who influenced them to purchase

Can we infer the underlying social network?

Inferring the Network

- There is a hidden directed network:



- We only see times when nodes get infected:
  - Cascade c_1: (a,1), (c,2), (b,3), (e,4)
  - Cascade c_2: (c,1), (a,4), (b,5), (d,6)

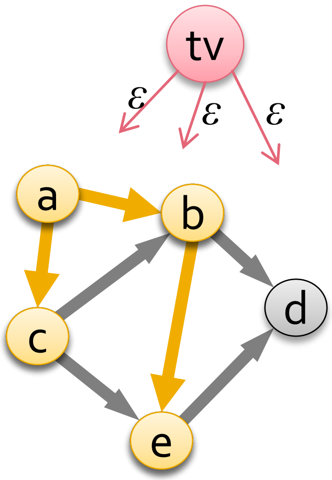
- Want to infer who-infects-whom network

Plan for the Talk

- The plan:
  - Define a continuous time model of diffusion
  - Define the likelihood of the observed data given a graph
  - Show how to efficiently compute the likelihood
  - Show how to efficiently optimize the likelihood
    - Find a graph G that maximizes the likelihood

Cascade generation model

- Cascade generation model:
  - Cascade reaches u at time t_u, and spreads to u's neighbors v:
    - With prob. β cascade propagates along (u,v) and $t_v = t_u + \Delta$, where $\Delta \sim f(\Theta)$



- Transmission probability:

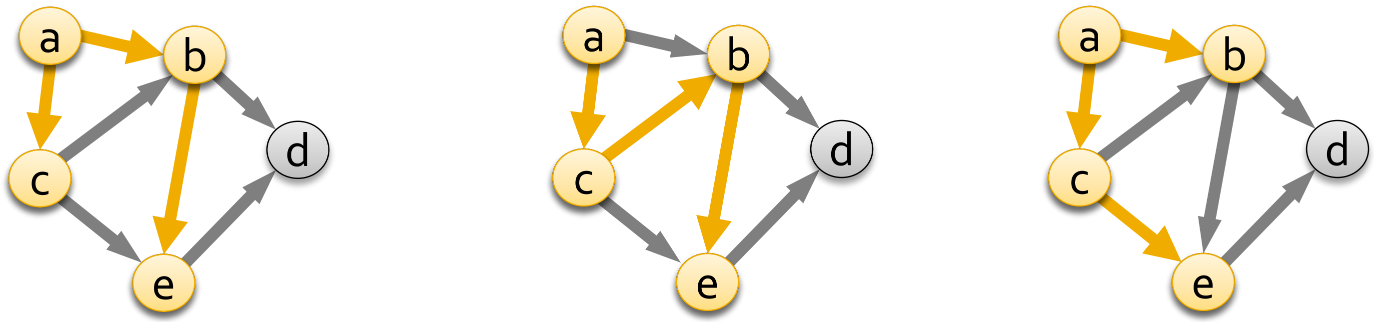
$$P_c(i,j) \propto P(\Delta) \text{ if } t_j > t_i \text{, else } \varepsilon$$

- Prob. that cascade c propagates in a tree T

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

Cascade generation model

- There are many possible transmission trees:
  - c: (a,1), (c,2), (b,3), (e,4)



- Heed to consider all possible directed spanning trees T supported by G:

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

Finding the Diffusion network

- Then simply: $P(C|G) = \prod_{c \in C} P(c|G)$
- Want to find: $G = \underset{|G| \leq k}{\operatorname{argmax}} P(C|G)$
- Good news: computing $P(C|G)$ is tractable
  - Need to consider all possible transmission trees of *G*
    - There are $O(n^n)$ such spanning trees!
  - The Matrix tree theorem
    - Can compute this sum in $O(n^3)$
- Bad news:
  - We actually want to find $\arg\max_G P(C|G)$

An alternative formulation

- Consider only the most likely tree
- Log-likelihood of a cascade c in graph G:

$T)$

- Log … cades C:

The problem is NP-hard:

MAX-k-COVER [KDD '10]

Our algorithm can do it near-optimally in $O(N^2)$

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

Good News

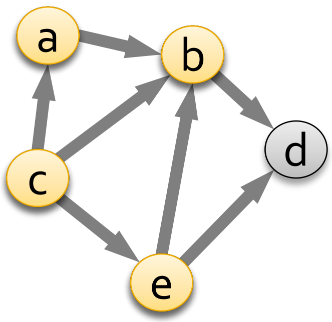
Given a cascade c

- What is the most likely propagation tree?

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

  - A maximum **directed** spanning tree
    - Edge *(i,j)* in G has weight $w(i,j) = \log P_c(i,j)$
    - To compute the maximum spanning tree:
      Each node just picks an in-edge of max weight

$$= \sum_{i \in V} \max_{Par_T(i)} w(Par_T(i), i)$$



Local greedy selection gives optimal tree!

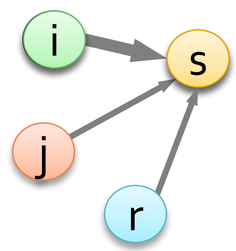
Great News

- <u>Theorem:</u>
  $F_C(G)$ is monotonic, and submodular

$$\underbrace{F_C(A \cup \{e\}) - F_C(A)}_{\text{Gain of adding an edge to a "small" graph}} \geq \underbrace{F_C(B \cup \{e\}) - F_C(B)}_{\text{Gain of adding an edge to a "large" graph}}$$

$$A \subseteq B \subseteq V \times V$$

- Proof:
  - Single cascade *c*, edge *e* of wgt. *x*
  - Let *w* be max weight in-edge of *s* in A
  - Let *w'* be max weight in-edge of *s* in B
  - We know: $w \leq w'$ and $x = x'$
  - Now: $F_c(A \cup \{e\}) - F_c(A) = \max(w, x) - w$
    $\geq \max(w', x) - w' = F_c(B \cup \{e\}) - F_c(B)$



Finding the graph

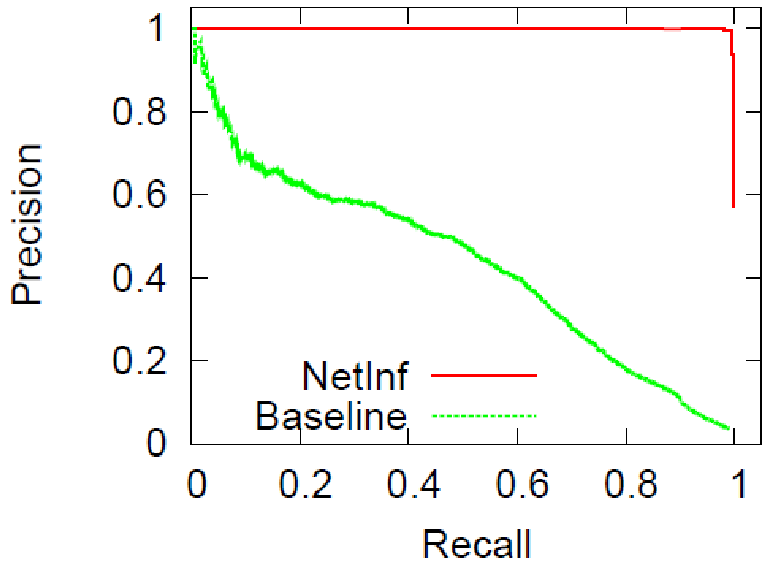
- Use the greedy hill-climbing to maximize $F_C(G)$:

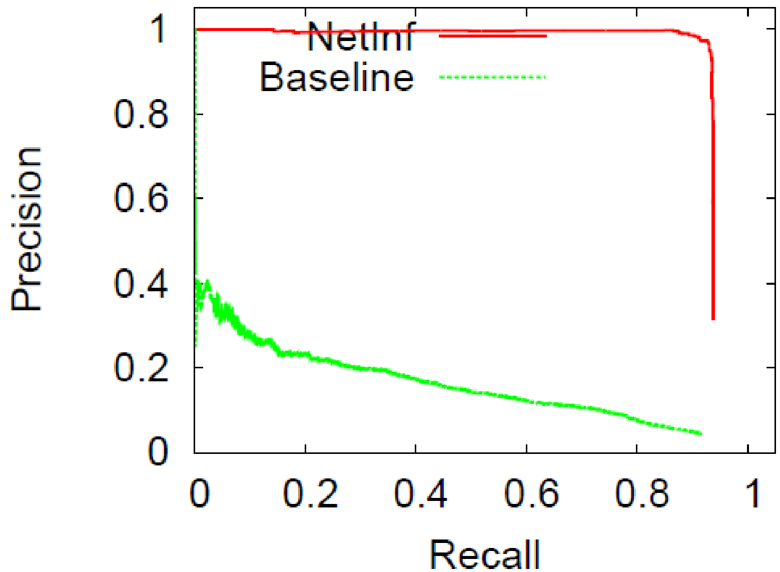
$$e_i = \underset{e \in G \setminus G_{i-1}}{\operatorname{argmax}} F_C(G_{i-1} \cup \{e\}) - F_C(G_{i-1})$$

  - At every step pick the edge that maximizes the marginal improvement
- Benefits:
  - Approximation guarantee (~0.63 of OPT)
  - Tight online bounds on the solution quality
  - Speed-ups:
    - Lazy evaluation
    - Localized update

Experimental setup

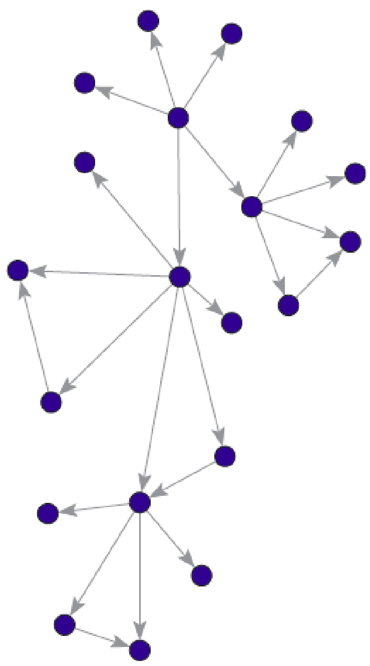
- Synthetic data:
  - Generate a graph G on *k* edges
  - Generate cascades
  - Record node infection times
  - Reconstruct G
- Evaluation:
  - How many edges of G can we find?
    - Precision-Recall
    - Break even point



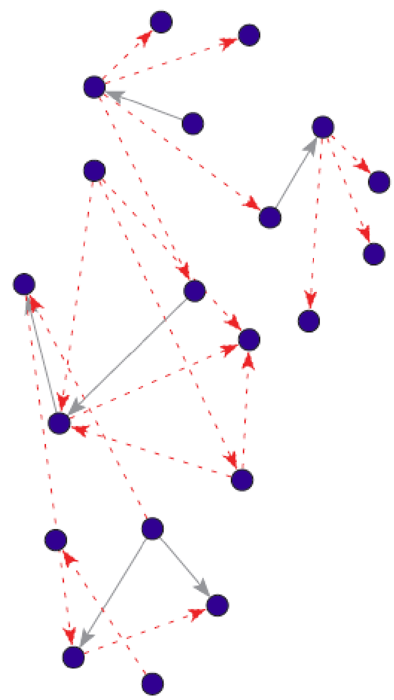


Small example

- Small synthetic network:



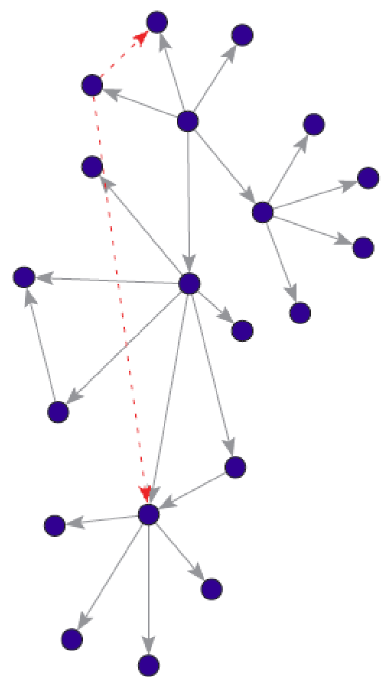
True network



Baseline network

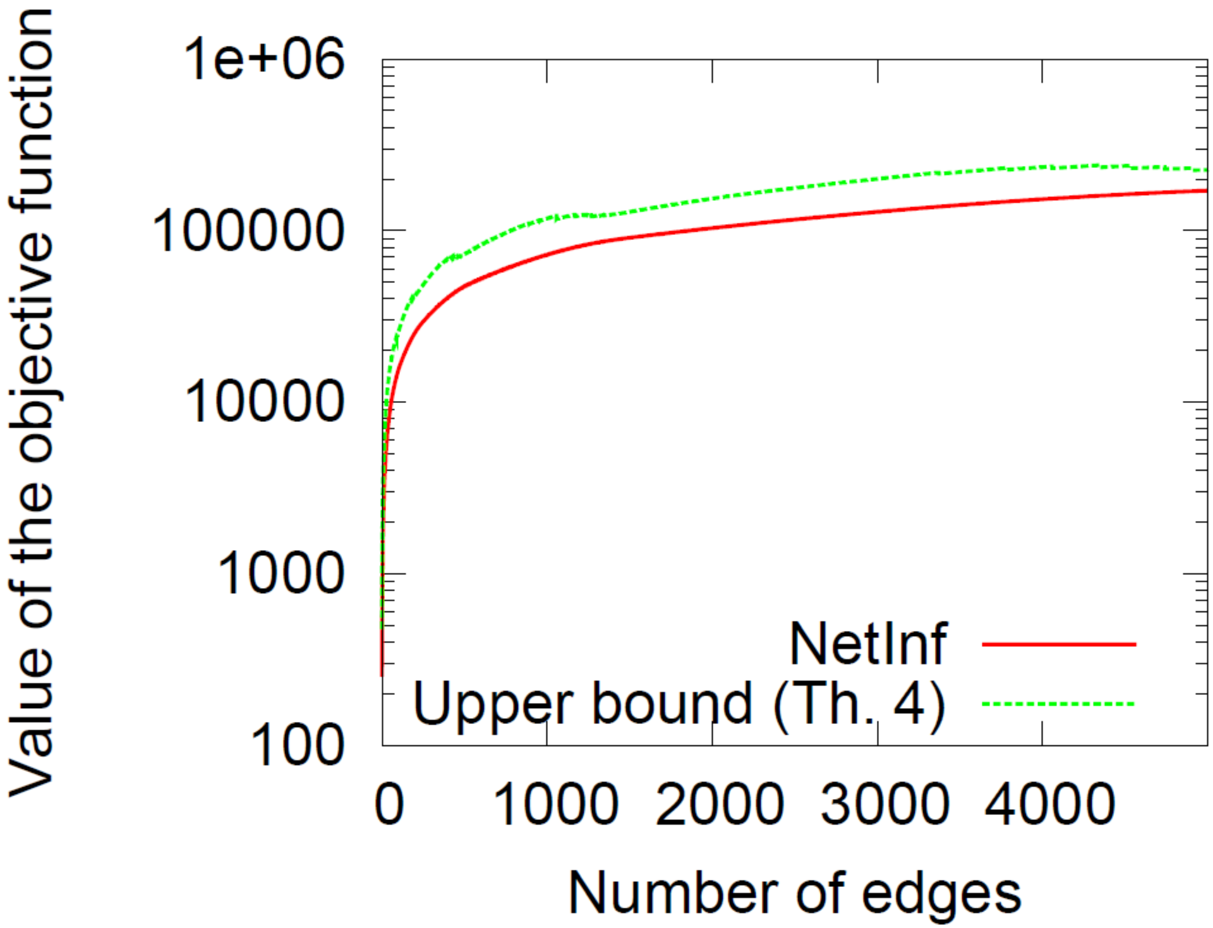
Pick strongest edges

$w(u,v) = \sum_{c \in C} P_c(u,v)$



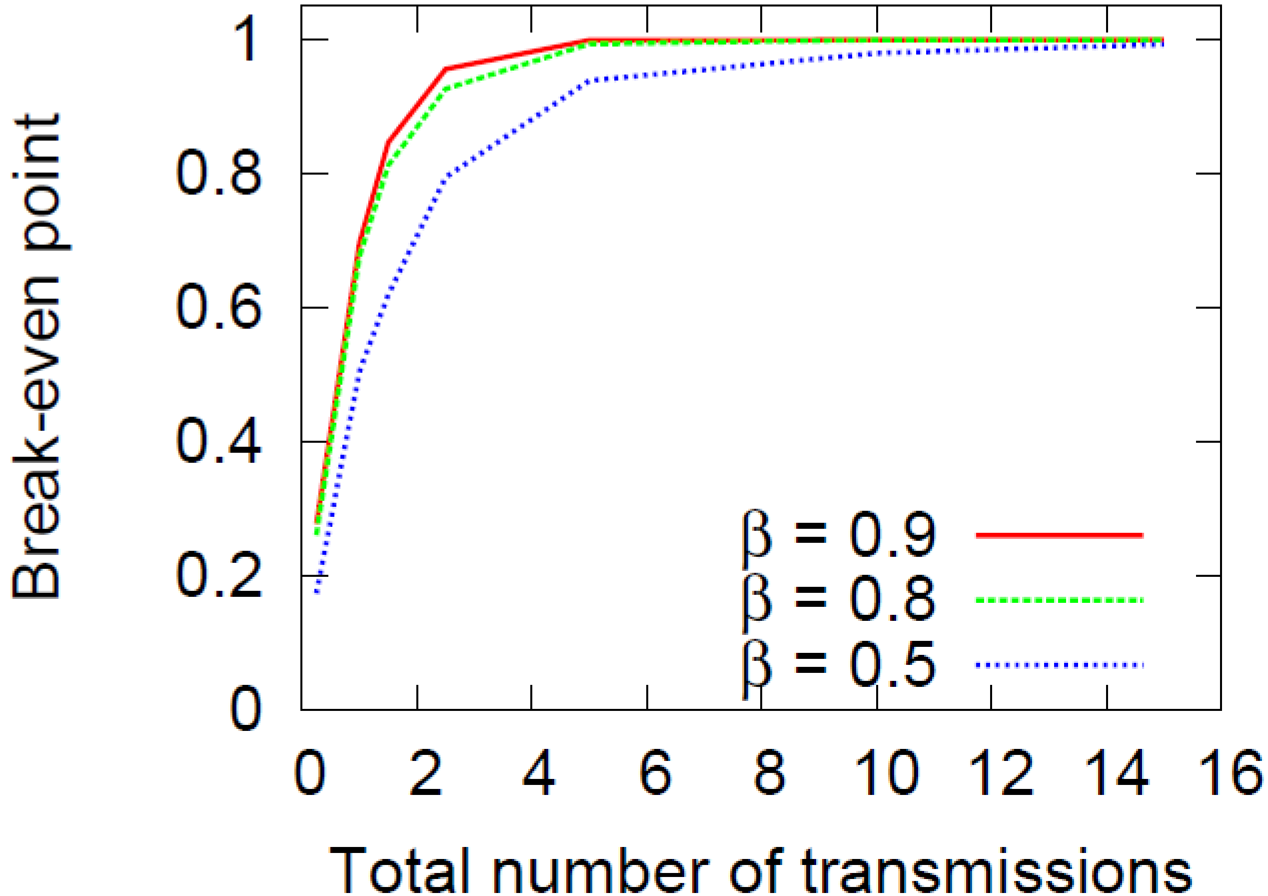
Our method

How well do we optimize $F_C(G)$



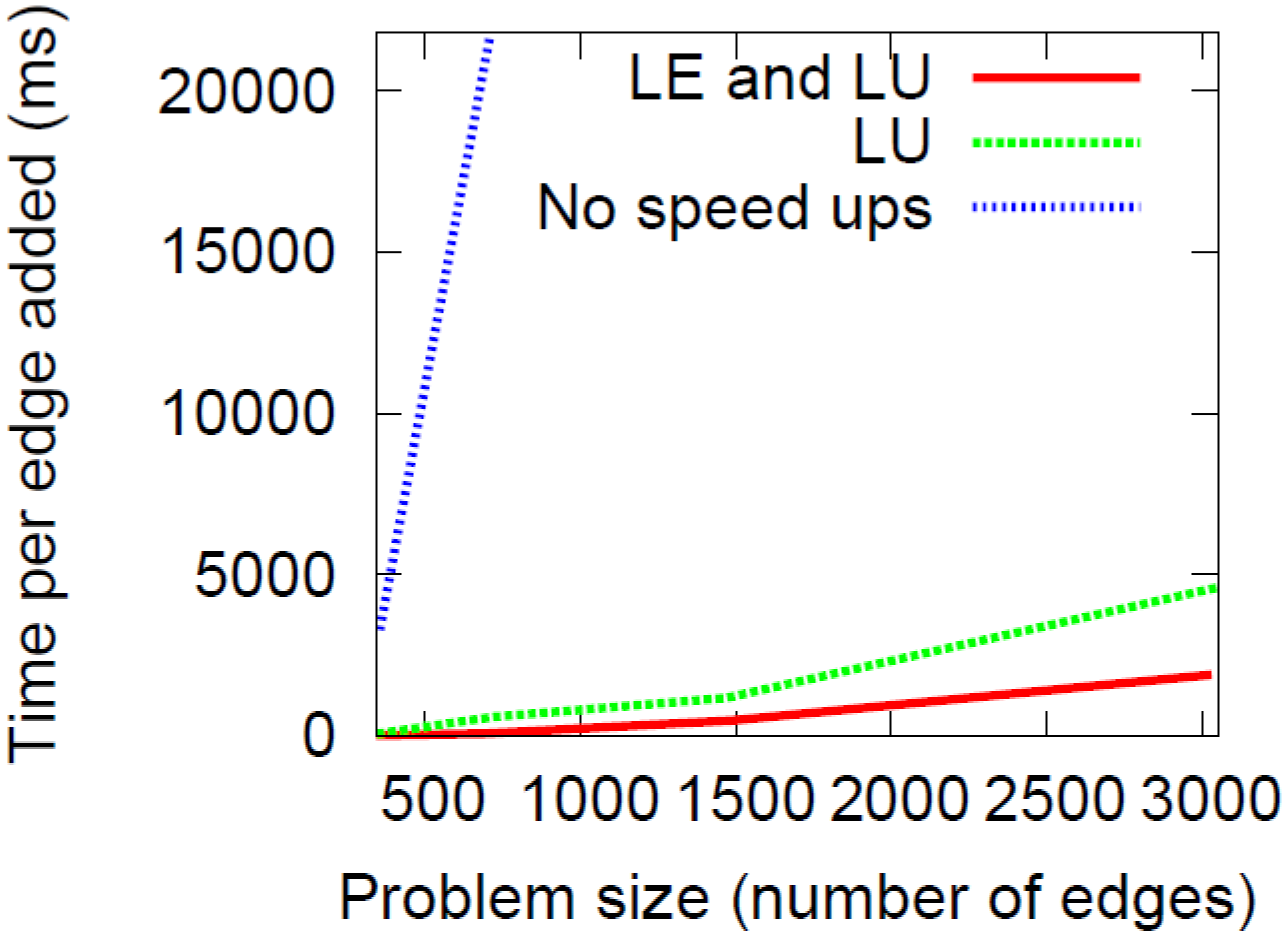
- Greedy hill-climbing gets inside 90% of OPT

How many cascades do we need?



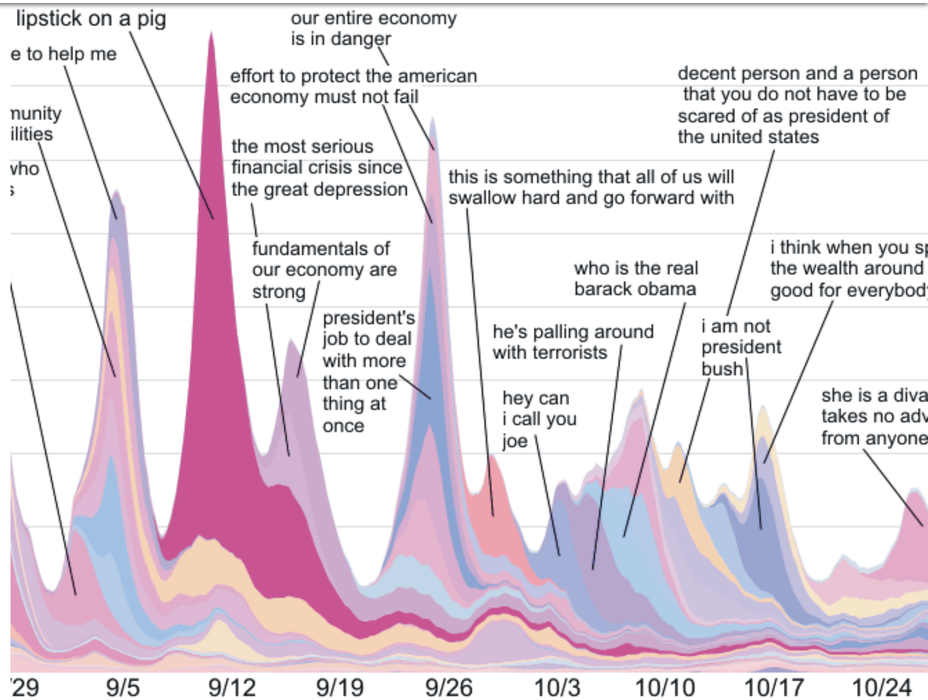
- With twice as many infections as edges the break-even point is at 0.8-0.9

Runtime


Time per edge added (ms)
20000
15000
10000
5000
0
500 1000 1500 2000 2500 3000
Problem size (number of edges)
LE and LU
LU
No speed ups

Experiments: Real data

- Memetracker dataset:
  - 172m news articles
  - Aug '08 – Sept '09
  - 343m textual phrases
  - Times $t_c(w)$ when site w mentions phrase c



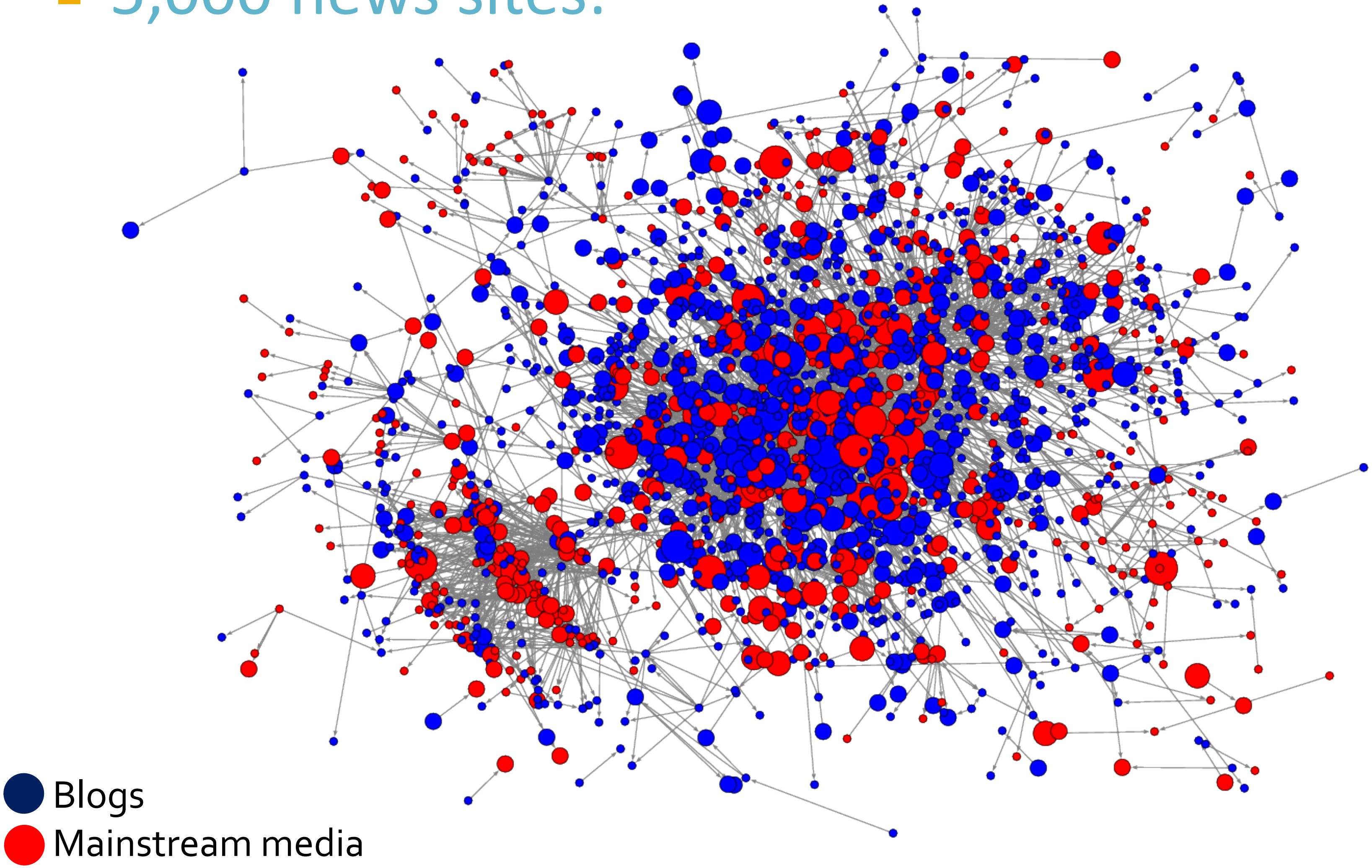
http://memetracker.org

- Given times when sites mention phrases
- Infer the network of information diffusion:
  - Who tends to copy (repeat after) whom

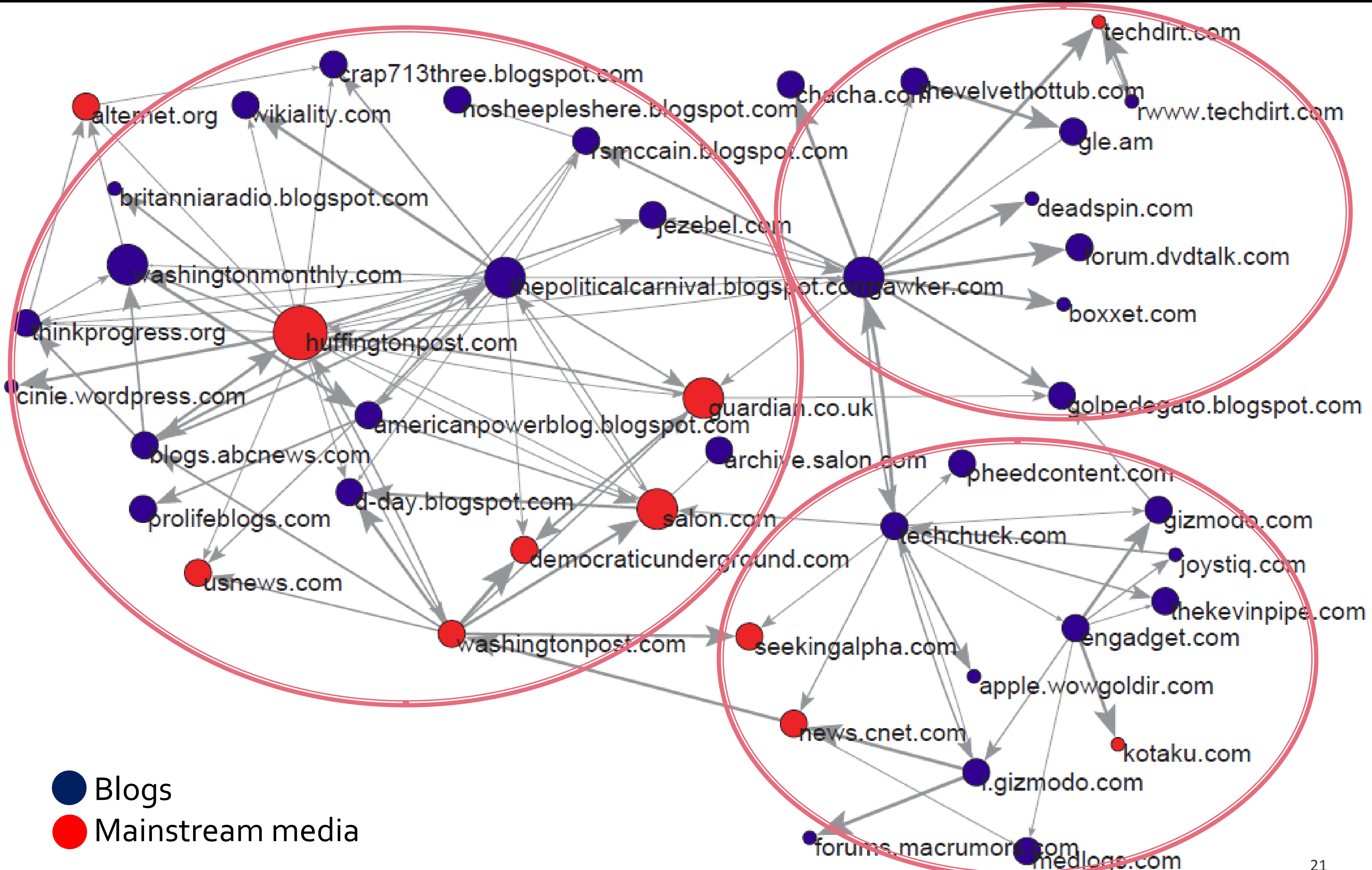
[w/ Gomez-Krause, '10]

Diffusion network

- 5,000 news sites:



Diffusion network (small part)



Conclusion

- Inferring hidden networks based on diffusion data
- Problem formulation in a maximum likelihood framework
  - Problem NP-hard in general
  - Developed an approximation algorithm that runs $O(N^2)$
- Future work:
  - Learn both the network and the diffusion model
  - Extensions to other processes taking place on networks

THANKS!

Data + Code:

http://snap.stanford.edu/netinf

Inferring Networks of Diffusion and Influence by M. Gomez-Rodriguez, J. Leskovec, A. Krause. *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 2010.

[Website] [Data]