Probabilistic Contagion in Graphs

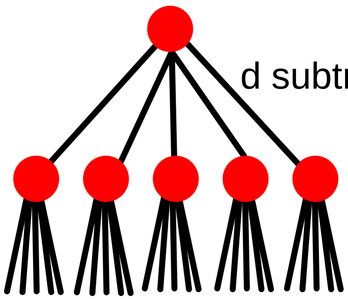
CS224W: Social and Information Network Analysis

Jure Leskovec, Stanford University

http://cs224w.stanford.edu

Probabilistic Spreading Models

- **Epidemic Model based on Random Trees**
  - **(a variant of branching processes)**
  - A patient meets *d* other people
  - With probability *q>0* infects each of them
- **Q: For which values of *d* and *q* does the epidemic run forever?**
  - Run forever: $\lim_{n\to\infty} P\begin{bmatrix} infected\ node \\ at\ depth\ n \end{bmatrix} > 0$
  - Die out: -- || -- $= 0$



Probabilistic Spreading Models

- p_n = prob. there is an infected node at depth *n*
- We need: $\lim_{n\to\infty} p_n = ?$ (based on *q* and *d*)
- **Need recurrence for p_n**

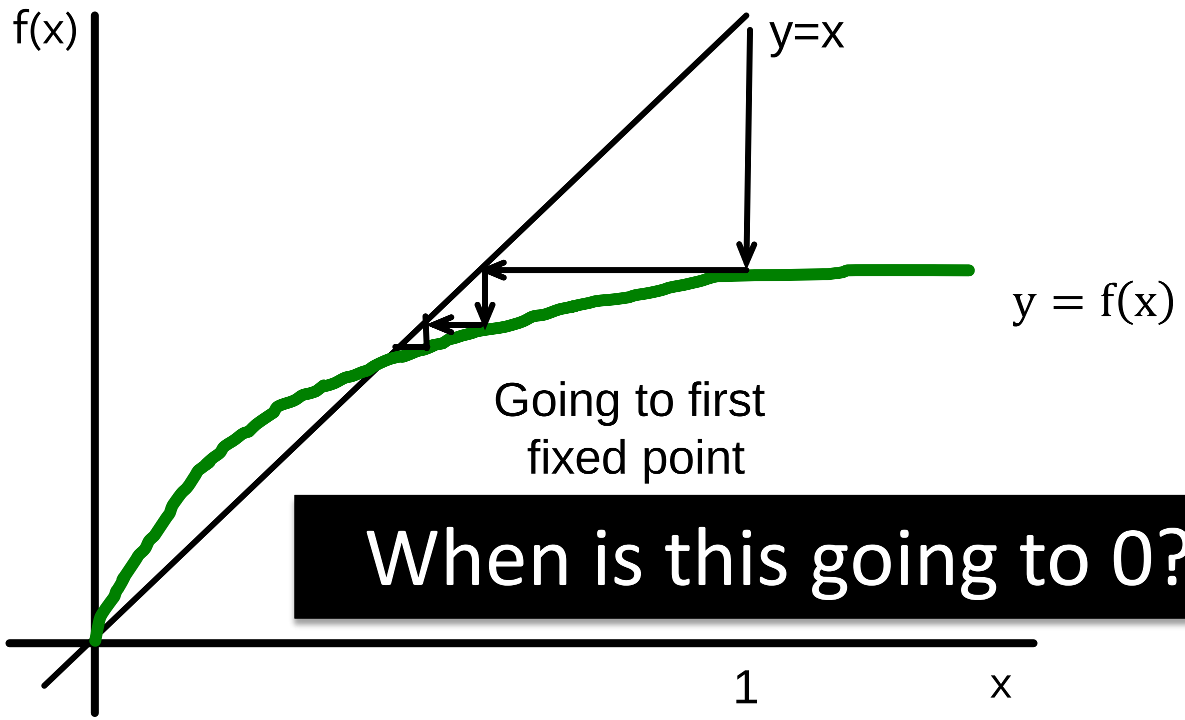
$$p_n = 1 - \underbrace{(1 - q p_{n-1})^d}_{\text{No infected node at depth n}}$$

- $\lim_{n\to\infty} p_n$ = result of iterating

$$\mathrm{f(x)} = 1 - (1 - qx)^d$$

  - Starting at x=1 (since p_1=1)

Fixed Point: $f(x) = 1 - (1 - qx)^d$



When is this going to 0?

What do we know about f(x)?

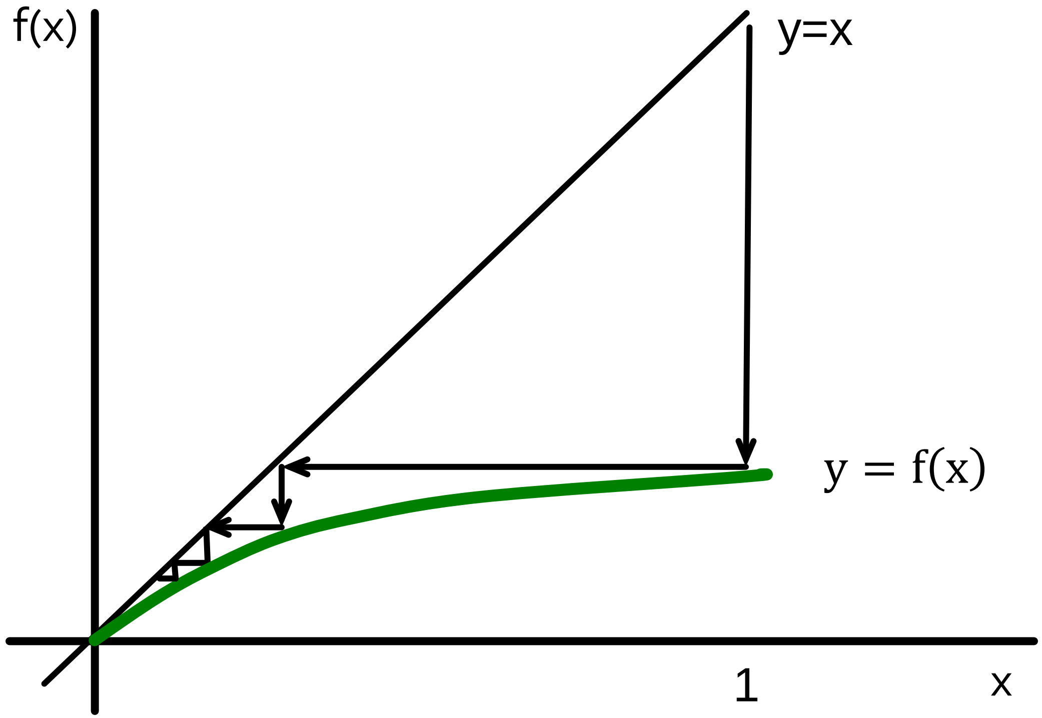
$f(0) = 0, f(1) = 1$

$f(1) = 1 - (1 - q)^d < 1$

$f'(x) = qd(1 - qx)^{d-1}$

$f'(0) = qd$: $f'(x)$ is monotone decreasing on [0,1]

Fixed Point: When is this zero?



We need f(x) to be bellow y=x!

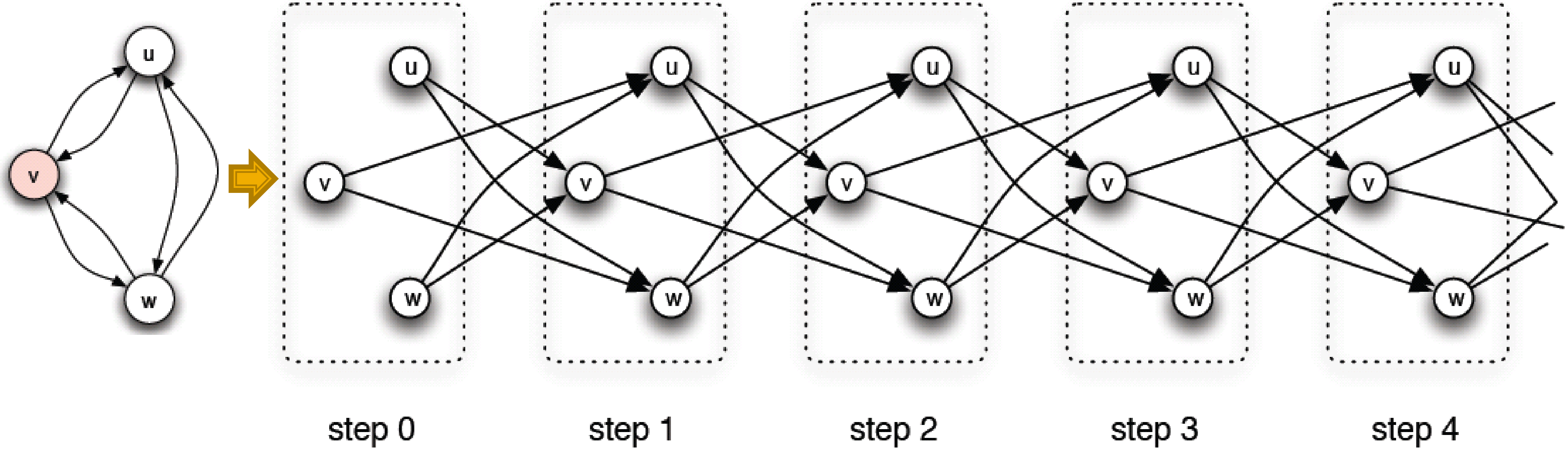
$$f'(0) < 1$$

$$\lim_{n\to\infty} p_n = 0\ ? \quad \text{to} \ \ qd < 1$$

qd = expected # of people at we infect

Probabilistic Contagion

- In this model nodes only go from inactive → active
- **Can generalize to allow nodes to alternate between active and inactive state by:**



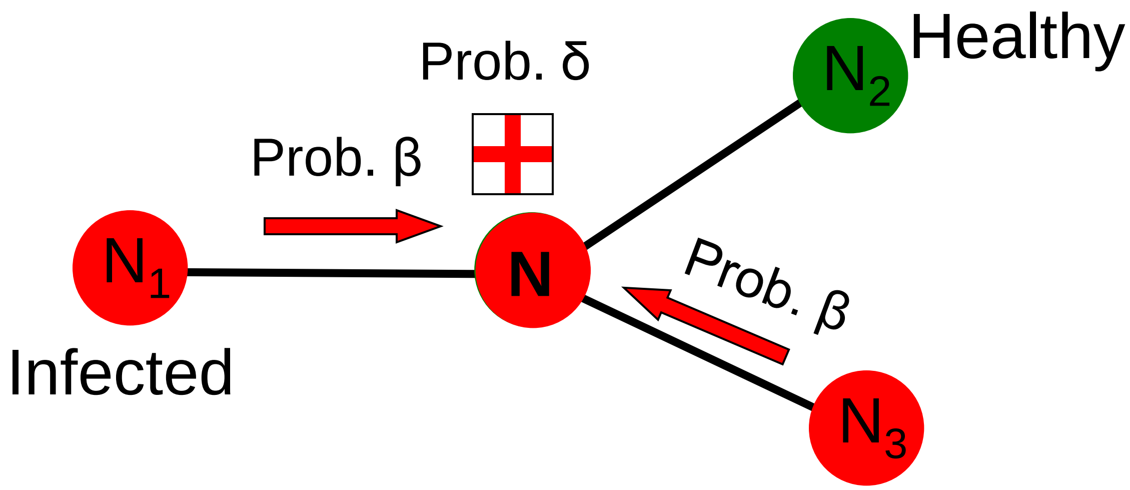
EXTRA:
Generalizing the Model to Virus Propagation

Spreading Models of Viruses

- Generalizing to model to **Virus Propagation**

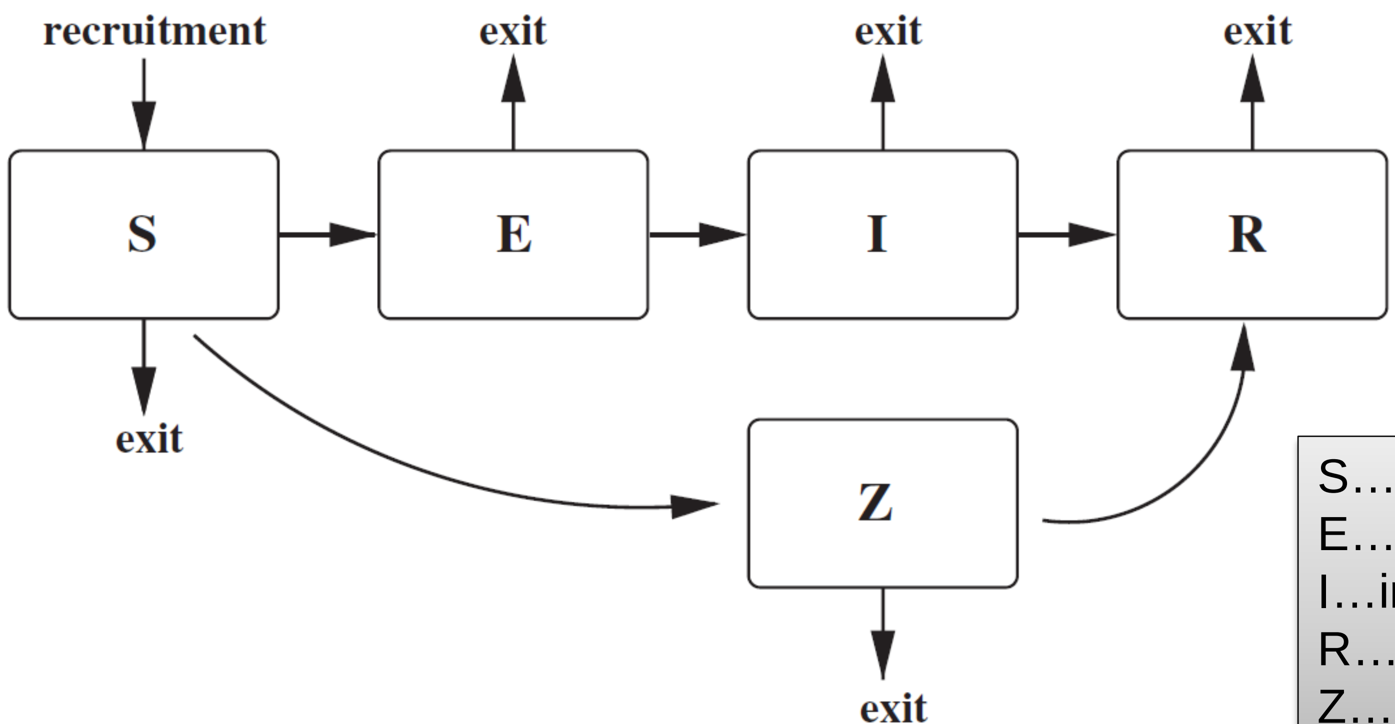
2 Parameters:

- (Virus) birth rate β:
  - probability than an infected neighbor attacks
- (Virus) death rate δ:
  - probability that an infected node heals



More Generally: S+E+I+R Models

- **General scheme for epidemic models:**
  - **Each node can go through phases:**
    - Transition probs. are governed by model parameters



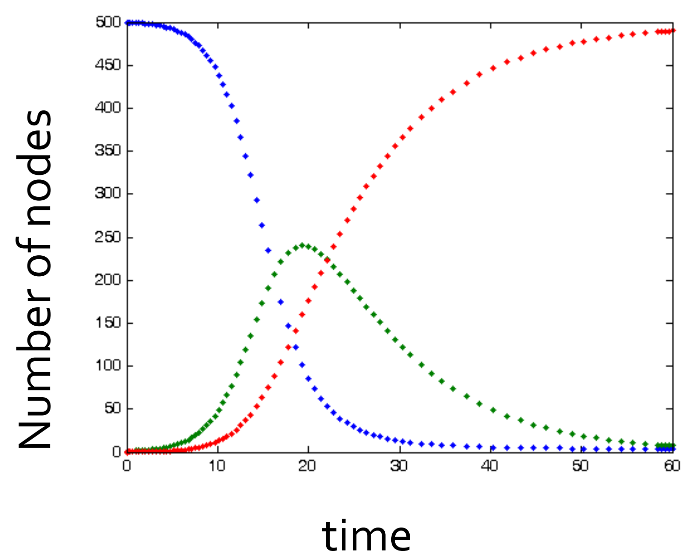
S…susceptible
E…exposed
I…infected
R…recovered
Z…immune

SIR Model

- Node goes through phases

  - Models chickenpox or plague:
    - Once you heal, you can never get infected again
- **Assuming perfect mixing**
  - network is a complete graph

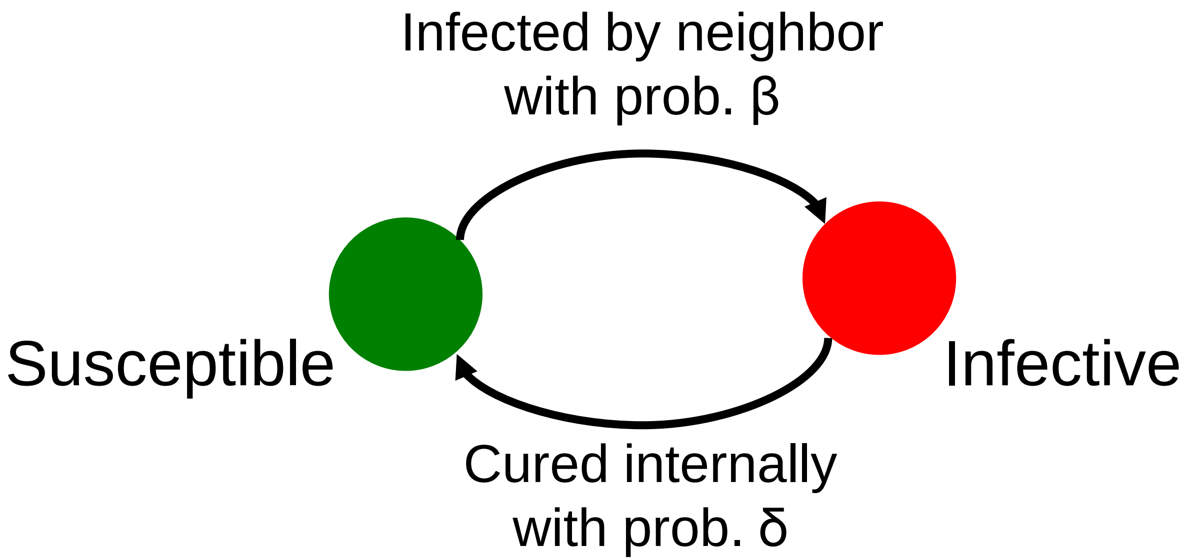
the model dynamics is



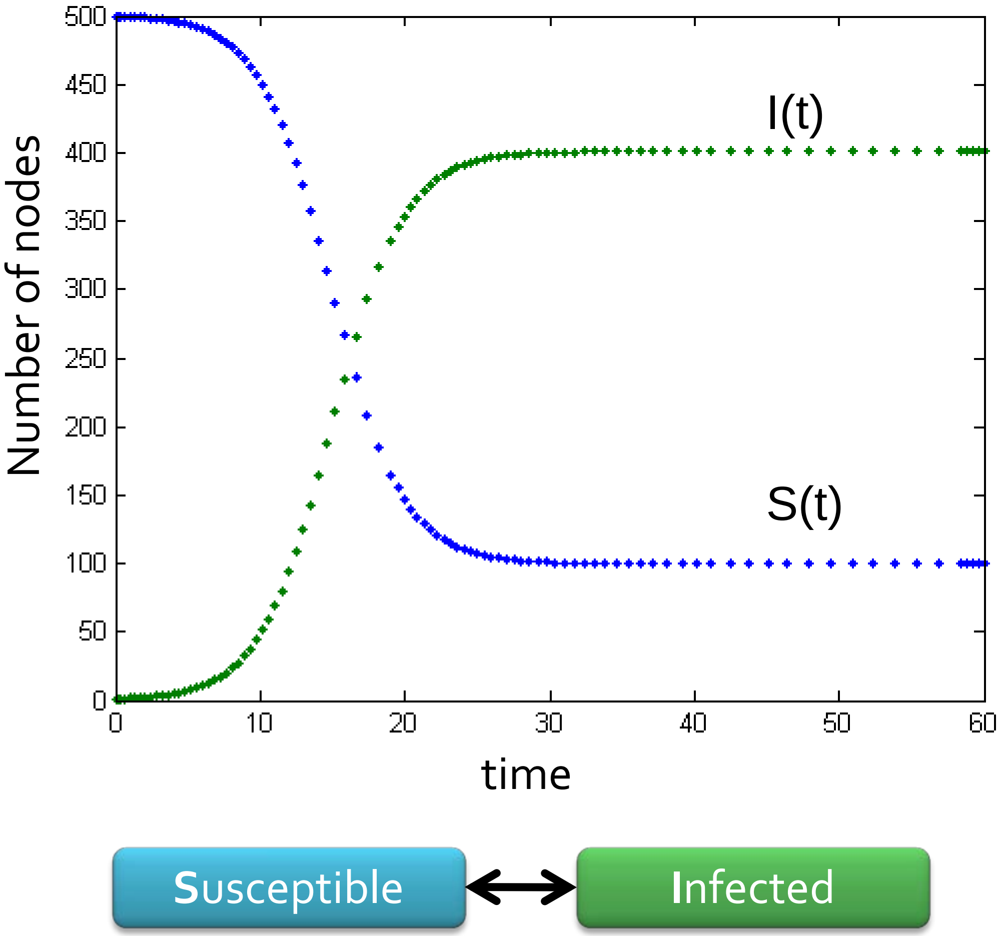
$$\frac{dS}{dt} = -\beta I S \qquad \frac{dI}{dt} = \beta I S - \nu I \qquad \frac{dR}{dt} = \nu I$$

SIS Model

- **Susceptible-Infective-Susceptible (SIS) model**
- Cured nodes immediately become susceptible
- **Virus "strength": s = β / δ**
- **Node state transition diagram:**



SIS Model



- **Models flu:**
  - Susceptible node becomes infected
  - The node then heals and become susceptible again
- **Assuming perfect mixing (complete graph):**

$$\frac{dS}{dt} = -\beta SI + \delta I$$

$$\frac{dI}{dt} = \beta SI - \delta I$$

Question: Epidemic threshold t

- **SIS Model**
- **Epidemic threshold** of a graph G is a value of t, such that:
  - **If virus strength $s = \beta / \delta < t$ the epidemic can not happen (it eventually dies out)**

- **Given a graph what is its epidemic threshold?**

Epidemic Threshold in SIS Model

- **We have no epidemic if:**

(Virus) Death rate

Epidemic threshold

$$\beta/\delta < \tau = 1/\ \lambda_{1,A}$$

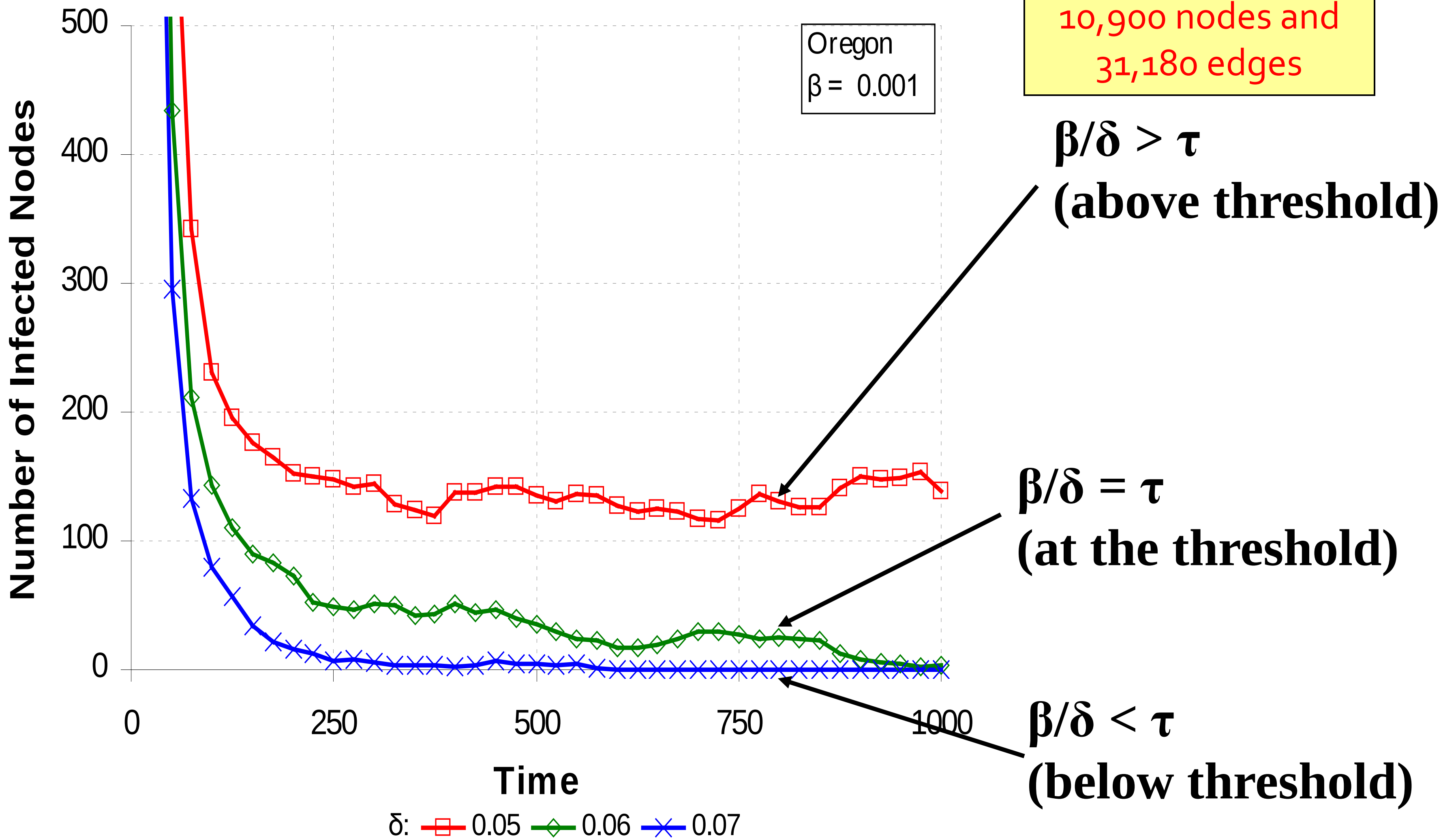
(Virus) Birth rate

largest eigenvalue of adj. matrix ***A***

► ***$\lambda_{1,A}$ alone captures the property of the graph!***

[Wang et al. 2003]

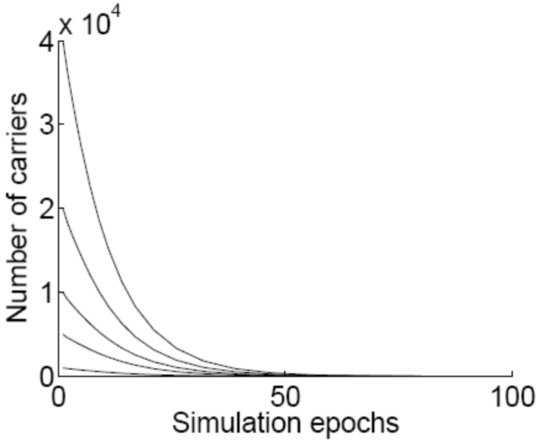
Experiments (AS graph)



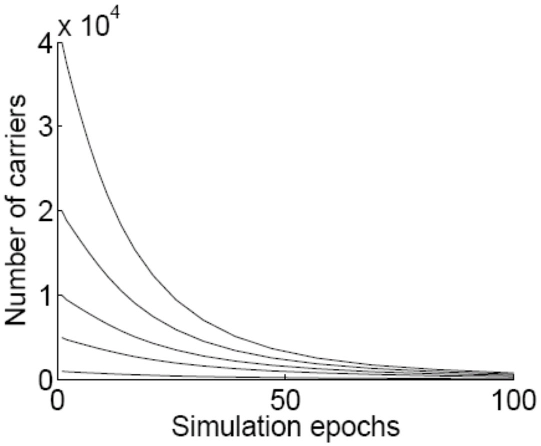
10,900 nodes and 31,180 edges

Experiments

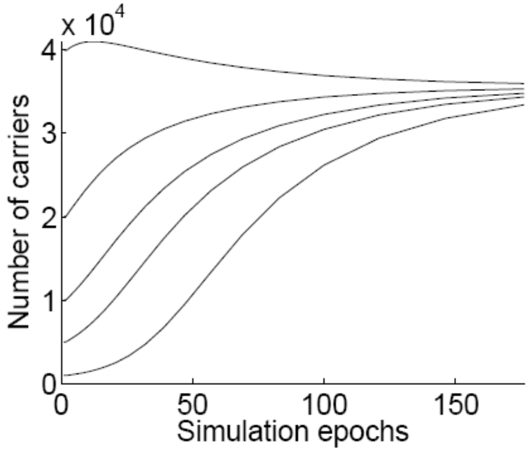
- **Does it matter how many people are initially infected?**



(a) Below the threshold, s=0.912



(b) At the threshold, s=1.003

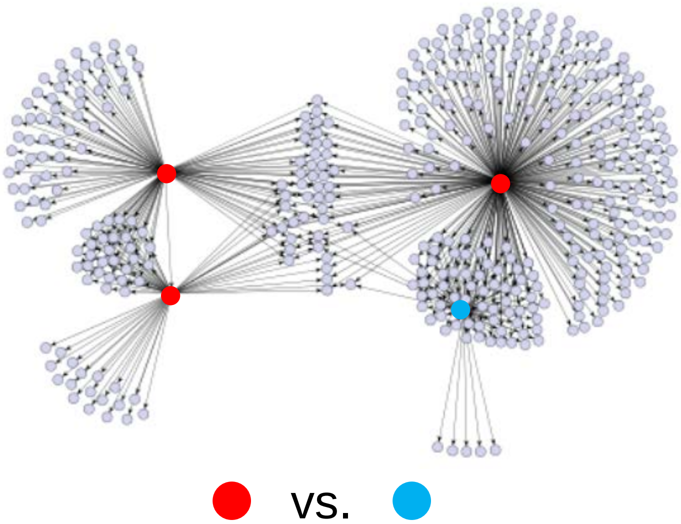


(c) Above the threshold, s=1.1

Influence Maximization in Graphs

How to Create Big Cascades?

- **Blogs – Information epidemics**
  - Which are the influential/infectious blogs?
  - Which blogs create big cascades?
- **Viral marketing**
  - Who are the influencers?
  - Where should I advertise?
- **Disease spreading**
  - Where to place monitoring stations to detect epidemics?

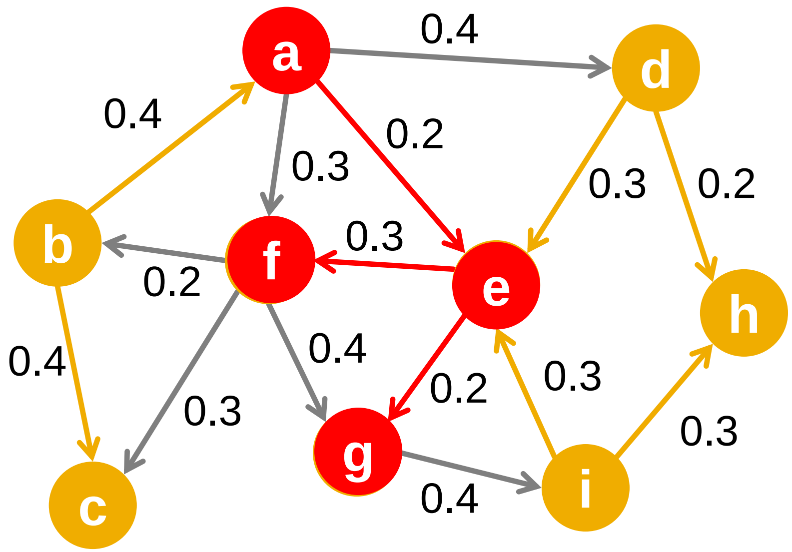


Probabilistic Contagion

- **Independent Cascade Model**
  - Directed finite G=(V,E)
  - Set *S* starts out with new behavior
    - Say nodes with this behavior are “active”
  - Each edge *(v,w)* has a probability p_{vw}
  - If node v is active, it gets one chance to make w active, with probability p_{vw}
    - Each edge fires at most once
- **Does scheduling matter?** **No**
    - E.g., u,v both active, doesn’t matter which fires first
  - **But the time moves in discrete steps**

Independent Cascade Model

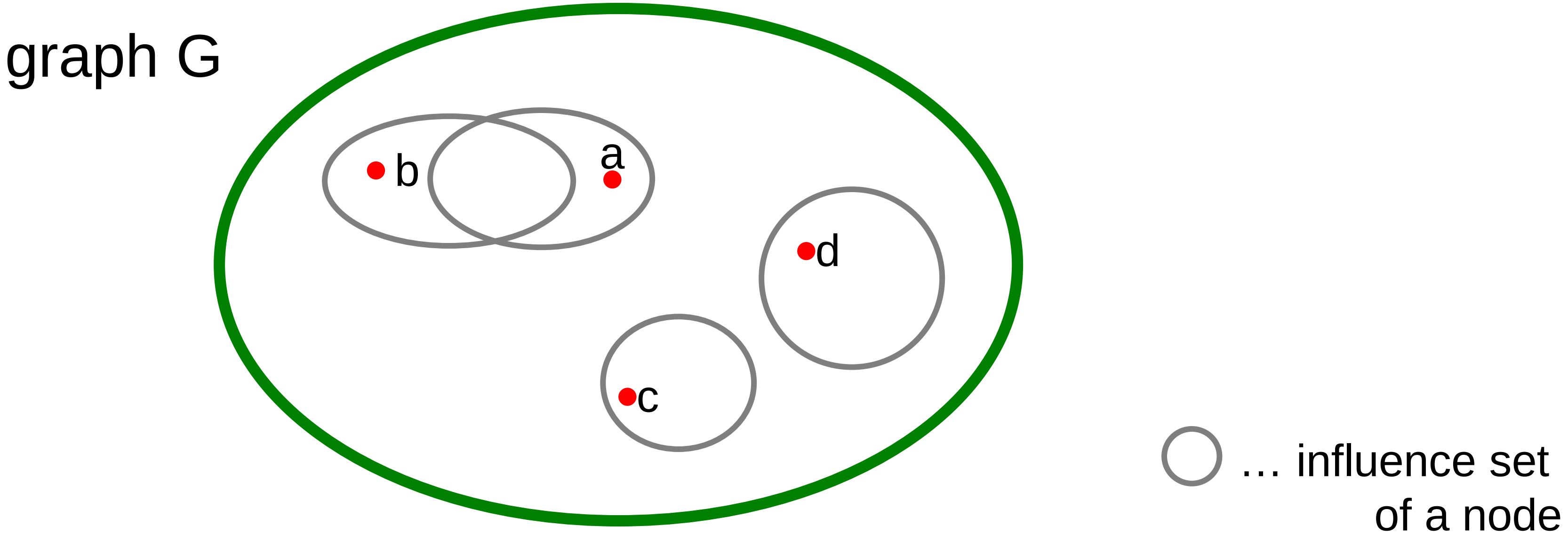
- Initially some nodes *S* are active
- Each edge *(v,w)* has probability (weight) p_{vw}



- When node v becomes active:
  - It activates each out-neighbor *w* with prob. p_{vw}
- Activations spread through the network

Most Influential Set of Nodes

- **S**: is initial active set
- ***f(S)***: the expected size of final active set



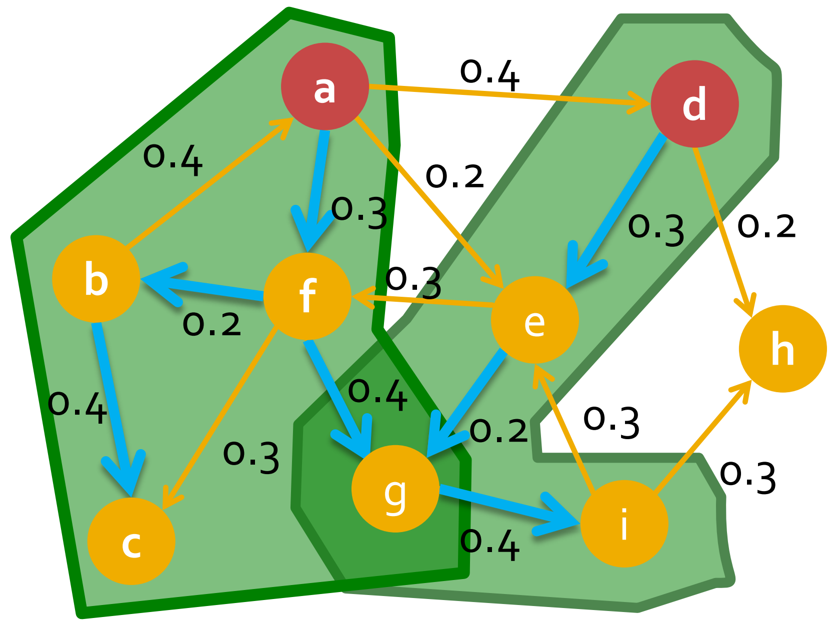
- **Set S is more influential if f(S) is larger**

  f({a,b} < f({a,c}) < f({a,d})

Most Influential Set

Problem:

- **Most influential set of size *k*:** set *S* of *k* nodes producing largest expected cascade size ***f(S)*** if activated [Domingos-Richardson '01]



Influence set of b

- **Optimization problem:** $\max_{S \text{ of size } k} f(S)$

Most Influential Subset of Nodes

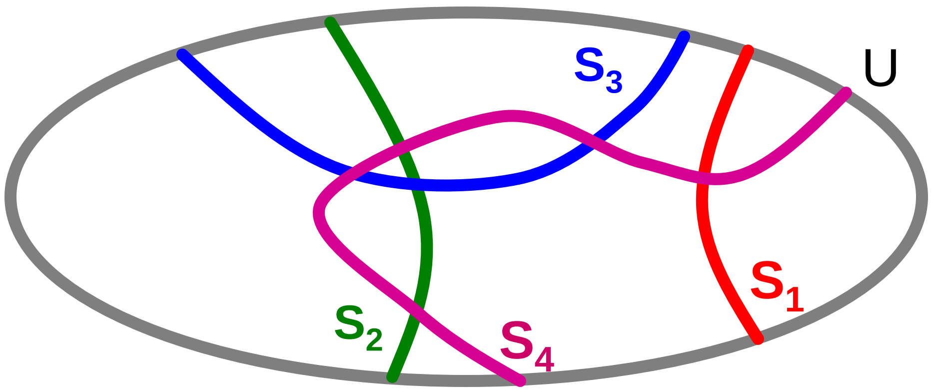
- **Most influential set of k nodes:** set *S* on *k* nodes producing largest expected cascade size *f(S)* if activated
- **The optimization problem:**

$$\max_{\text{S of size k}} f(S)$$

- **How hard is this problem?**
  - **NP-HARD!**
    - Show that finding most influential set is at least as hard as a vertex cover

Background: Vertex Cover

- **Vertex cover problem:**
  - Given universe of elements $U=\{u_1,\ldots,u_n\}$ and sets $S_1,\ldots, S_m \subseteq U$
  - **Are there k sets among $S_1,\ldots, S_m$ such that their union is U?**

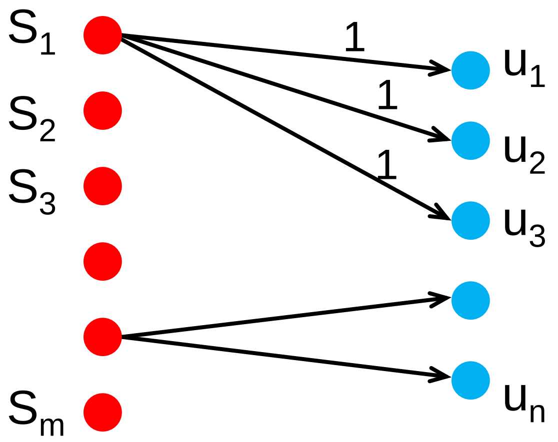


- **Goal:**

Encode vertex cover as an instance of $\max_{\text{S of size k}} f(S)$

Influence Maximization is NP-hard

- **Given a vertex cover instance with sets $S_1,\ldots, S_m$**
- **Build a bipartite "S-to-U" graph:**



e.g.:
$S_1=\{u_1, u_2, u_3\}$

Construction:
- Create edge (S_i,u) $\forall S_i$ $\forall$ $u \in S_i$ -- directed edge from sets to their elements
- Put weight 1 on each edge

- **There exists a set *S* of size *k* with *f(S)=k+n* iff there exists a size *k* set cover**

Note: Optimal solution is always a set of S_i
This is hard in general, could be special cases that are easier

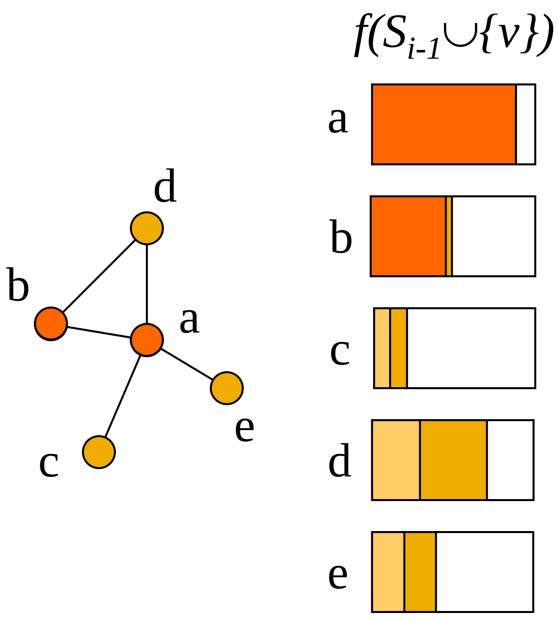
Summary so Far

- **Bad news:**
  - **Influence maximization is NP-hard**
- **Next, good news:**
  - **There exists an approximation algorithm!**
- **Consider the Hill Climbing algorithm to find S:**
  - **Input:** Influence set of each node u = $\{v_1, v_2, \dots\}$
    - If we activate u, nodes $\{v_1, v_2, \dots\}$ will eventually get active
  - **Algorithm:** At each step take the node u that gives best marginal gain: $\max f(S_{i-1} \cup \{u\})$

(Greedy) Hill Climbing

Algorithm:

- Start with $S_0=\{\}$
- For $i=1\ldots k$
  - Take node v that max $f(S_{i-1} \cup \{v\})$
  - Let $S_i = S_{i-1} \cup \{v\}$
- **Example:**
  - Eval f({a}),… f({d}), pick max
  - Eval f({a,b}),… f({a,d}), pick max
  - Eval f(a,b,c},… f({a,b,d}, pick …



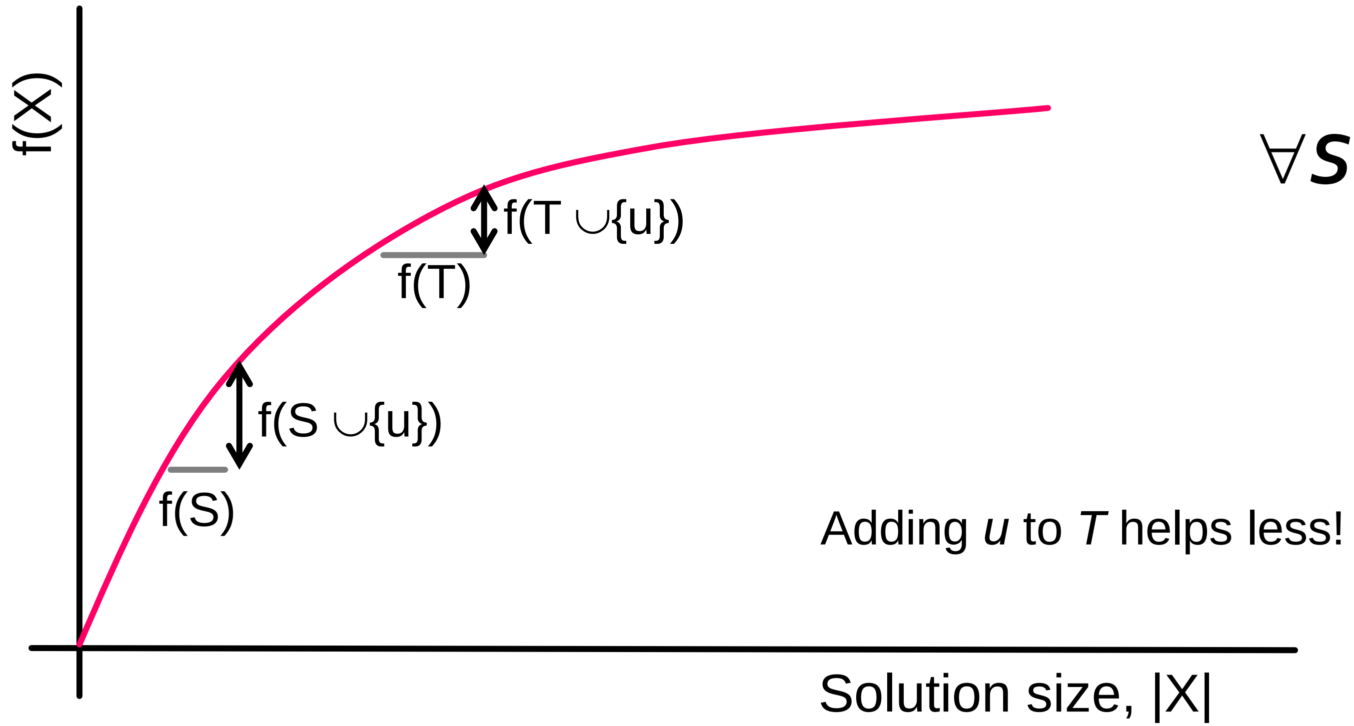
Approximation Guarantee

- **Hill climbing produces a solution S where: f(S) ≥(1-1/e)*OPT** (f(S)>0.63*OPT)
  [Nemhauser, Fisher, Wolsey '78, Kempe, Kleinberg, Tardos '03]
- **Claim holds for functions *f()* with 2 properties:**
  - ***f* is monotone:** (activating more nodes doesn't hurt)
    if $S \subseteq T$ then $f(S) \leq f(T)$ and $f(\{\})=0$
  - ***f* is submodular:** (activating each additional node helps less)
    adding an element to a set gives less improvement than adding it to one of its subsets: $\forall S \subseteq T$

$$\underbrace{f(S \cup \{u\}) - f(S)}_{\text{Gain of adding a node to a small set}} \geq \underbrace{f(T \cup \{u\}) - f(T)}_{\text{Gain of adding a node to a large set}}$$

Submodularity– Diminishing returns

- **Diminishing returns:**



$$\forall S \subseteq T$$

$$\underbrace{f(S \cup \{u\}) - f(S)}_{\text{Gain of adding a node to a small set}} \geq \underbrace{f(T \cup \{u\}) - f(T)}_{\text{Gain of adding a node to a large set}}$$

Plan: Prove 2 things

(1) Our f(S) is submodular

(2) Hill Climbing gives near-optimal solutions

(for monotone submodular functions)

Background: Submodular Functions

- **We must show our *f()* is submodular:**
- $\forall S \subseteq T$ (trivially $u \notin T$)

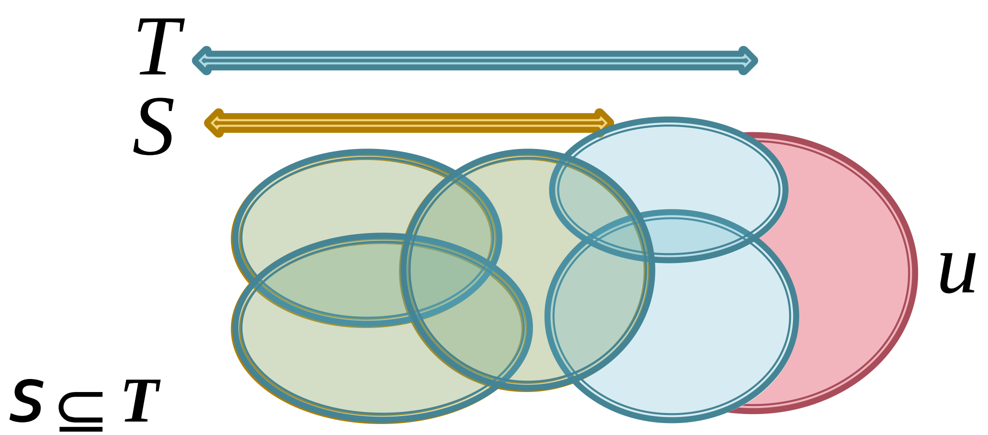
$$\underbrace{f(S \cup \{u\}) - f(S)}_{\text{Gain of adding a node to a small set}} \geq \underbrace{f(T \cup \{u\}) - f(T)}_{\text{Gain of adding a node to a large set}}$$

- **Basic fact 1:**
  - If $f_1(x), \ldots, f_k(x)$ are **submodular**, and $c_1, \ldots, c_k \geq 0$ then $F(x) = \sum_i c_i \cdot f_i(x)$ is also **submodular**

    (Linear combination of submodular functions is a submodular function)

Background: Submodular Functions

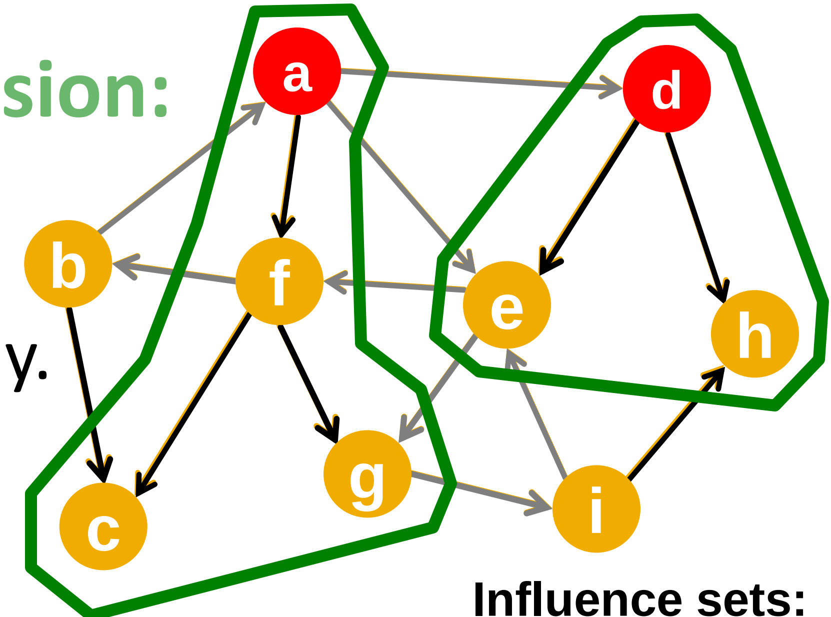
- $\forall S \subseteq T: \underbrace{f(S \cup \{u\}) - f(S)}_{\text{Gain of adding } u \text{ to a small set}} \geq \underbrace{f(T \cup \{u\}) - f(T)}_{\text{Gain of adding } u \text{ to a large set}}$
- **Basic fact 2: A simple submodular function**
  - Sets $A_1, \ldots, A_m$
  - $f(S) = |\bigcup_{i \in S} A_i|$ **(size of the union of sets A_i, $i \in S$)**
  - **Claim: f(S) is submodular!**



The more sets you already have the less new area a new set will cover

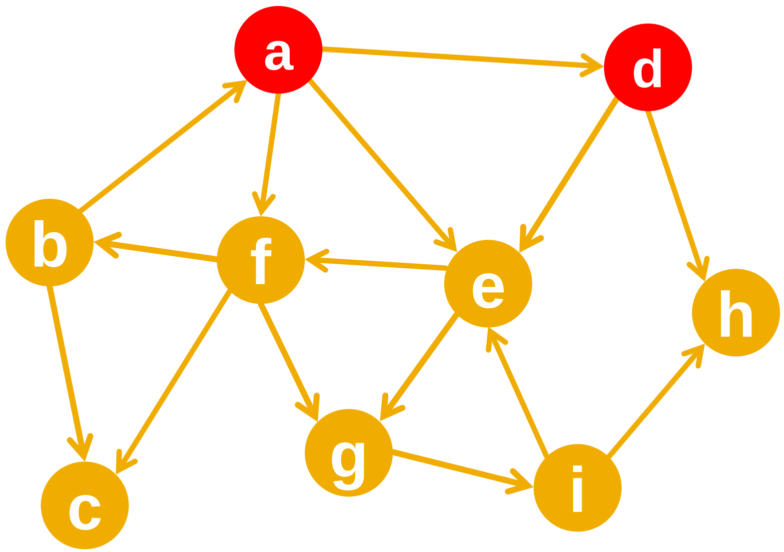
Our *f(S)* is Submodular!

- **Principle of deferred decision:**
  - Flip all the coins at the beginning and record which edges fire successfully.
  - **Now we have a deterministic graph!**
  - Edges which succeed are **live**
- **For the i-th realization of coin flips**
  - ***$f_i(S)$*** **=** size of the set reachable by live-edge paths from nodes in S
    - $f_i(S=\{a,b\}) = \{a,f,c,g,b\}$
    - $f_i(S=\{a,d\}) = \{a,f,c,g,d,e,h\}$



Influence sets:
$f_i(a) = \{a,f,c,g\}$
$f_i(d) = \{d,e,h\}$
$f_i(b) = \{b,c\}, \ldots$

Our $f(S)$ is Submodular!



- **Fix outcome *i* of coin flips**
- $f_i(v)$ = set of nodes reachable from v on live-edge paths
- $f_i(S)$ = size of cascades from S given coin flips i
- $f_i(S) = |\bigcup_{v \in S} f_i(v)| \Rightarrow f_i(S)$ is **submodular**
  - $f_i(v)$ are sets and $f_i(S)$ is the size of the union
- **Expected influence set size:**

  $f(S) = \sum_i f_i(S) \Rightarrow f(S)$ is **submodular!**
  - $f(S)$ is linear combination of submodular functions

Plan: Prove 2 things

(1) Our f(S) is submodular

(2) Hill Climbing gives near-optimal solutions

(for monotone submodular functions)

Proof for Hill Climbing

<u>**Claim:**</u>

If f(S) is monotone and submodular.
Hill climbing produces a solution S
where: **$f(S) \geq (1\text{-}1/e)*OPT$** $(f(S)>0.63*OPT)$

- **Setting**
  - Keep adding nodes that give the largest gain
  - Start with $S_0=\{\}$, produce sets $S_1, S_2,\ldots,S_k$
  - **Add elements one by one**
  - **Marginal gain:** $\delta_i = f(S_i) - f(S_{i-1})$
  - Let $T=\{t_1\ldots t_k\}$ be **the optimal set** of size k
- **We need to show: $f(S) \geq (1\text{-}1/e)\, f(T)$**

Basic Hill Climbing Fact

- $f(A \cup B) - f(A) \leq \sum_{j=1}^{k} [f(A \cup \{b_j\}) - f(A)]$
  - **where:** $B = \{b_1,\ldots,b_k\}$ and f is submodular,
- **Proof:**
  - Let $B_i = \{b_1,\ldots b_i\}$, so we have $B_1, B_2, \ldots, B_k = B$
  - $f(A \cup \mathrm{B}) - f(A) = \sum_{i=1}^{k} f(A \cup B_i) - f(A \cup B_{i-1})$
  - $= \sum_{i=1}^{k} f(A \cup B_{i-1} \cup \{b_i\}) - f(A \cup B_{i-1})$
  - $\leq \sum_{i=1}^{k} f(A \cup \{b_i\}) - f(A)$

By submodularity
since A∪X ∪{b} ⊇ A∪{b}

Work out the sum.
Everything but 1st and last term cancels out.

$f(A \cup B_1) - f(A \cup B_0)$
$+ f(A \cup B_2) - f(A \cup B_1)$
$+ f(A \cup B_3) - \cdots$
$+ f(A \cup B_k) - f(A \cup B_{k-1})$

What is δ_i (e.i., Gain in step i)?

- $f(T) \le f(S_i \cup T)$ (by monotonicity)
- $= \underbrace{f(S_i \cup T) - f(S_i)} + f(S_i)$
- $\le \sum_{j=1}^{k} \left[f\left(S_i \cup \{t_j\}\right) - f(S_i) \right] + f(S_i)$ (by prev. slide)
- $\le \sum_{j=1}^{k} \delta_{i+1} + f(S_i) = f(S_i) + k\,\delta_{i+1}$
- **Thus:** $f(T) \le f(S_i) + k\,\delta_{i+1}$
- $\Rightarrow \delta_{i+1} \ge \frac{1}{k}\left[f(T) - f(S_i) \right]$

$T = \{t_1, \ldots t_k\}$
t_j is one choice of a next element, and we greedily choose **the best one**, for a gain of δ_{i+1}

What is f(S_{i+1})?

- **We just showed:** $\delta_{i+1} \geq \frac{1}{k}[f(T) - f(S_i)]$
- **What is f(S_{i+1})?**
  - $f(S_{i+1}) = f(S_i) + \delta_{i+1}$
  - $\geq f(S_i) + \frac{1}{k}[f(T) - f(S_i)]$
  - $= \left(1 - \frac{1}{k}\right) f(S_i) + \frac{1}{k} f(T)$
- **What is f(S_k)?**

What is $f(S_k)$?

- Claim: $f(S_i) \geq \left[1-\left(1-\frac{1}{k}\right)^i\right] f(T)$

Proof by induction:

- $i = 0$:
  - $f(S_0) = f(\{\}) = 0$
  - $\left[1 - \left(1 - \frac{1}{k}\right)^0\right] f(T) = 0$

What is f(S$_k$)?

- <u>Claim:</u> $f(S_i) \geq \left[1 - \left(1 - \frac{1}{k}\right)^i\right] f(T)$

Proof by induction:

- At $i+1$:
  - $f(S_{i+1}) \geq \left(1 - \frac{1}{k}\right) f(S_i) + \frac{1}{k} f(T)$
  - $\geq \left(1 - \frac{1}{k}\right) \left[1 - \left(1 - \frac{1}{k}\right)^i\right] f(T) + \frac{1}{k} f(T)$
  - $= \left[1 - \left(1 - \frac{1}{k}\right)^{i+1}\right] f(T)$

What is f(S_k)?

- **Thus:**

$$f(S) = f(S_k) \geq \left[1 - \underbrace{\left(1 - \frac{1}{k}\right)^k}_{\leq \frac{1}{e}}\right] f(T)$$

- **Then:**

$$f(S_k) \geq \left(1 - \frac{1}{e}\right) f(T)$$

qed

Solution Quality

We just proved:

- Hill climbing finds solution *S* which **f(S) ≥ (1-1/e)*OPT**
  - this is a **data independent bound**
    - This is a worst case bound
    - No matter what is the input data (influence sets) we know that Hill Climbing won't do worse than 0.63*OPT

Data dependent bound:

- We want a bound whose value depends on the input data

  If the data is "easy", we are likely doing better than 63% of OPT

Data Dependent Bound

- Suppose S is some solution to

  $\text{argmax}_S f(S)$ s.t. $|S| \leq k$

  f() is monotone & submodular

  and let $T = \{t_1,\ldots,t_k\}$ be the OPT solution
- **CLAIM:**

  For each $u \notin S$ let $\delta_u = f(S \cup \{u\}) - f(S)$

  Order δ_u so that $\delta_1 \geq \delta_2 \geq \ldots \geq \delta_n$

  Then: $f(T) \leq f(S) + \sum_{i=1}^{k} \delta_i$

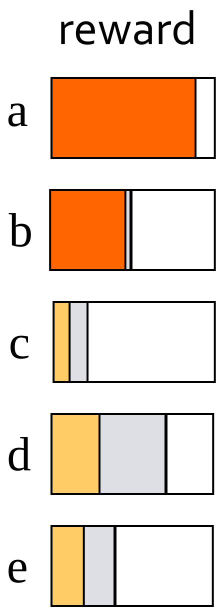
Data Dependent Bound

- For each $u \notin S$ let $\delta_u = f(S \cup \{u\}) - f(S)$
  Order δ_u so that $\delta_1 \geq \delta_2 \geq \ldots \geq \delta_n$
  Then: $f(T) \leq f(S) + \sum_{i=1}^{k} \delta_i$
- **Proof:**
  - $f(T) \leq f(T \cup S) =$
    $f(S) +$
    $\sum_{i=1}^{k} [f(S \cup \{t_1 \ldots t_i\}) - f(S \cup \{t_1 \ldots t_{i-1}\})]$
  - $\leq f(S) + \sum_{i=1}^{k} [f(S \cup \{t_i\}) - f(S)]$
  - $= f(S) + \sum_{i=1}^{k} \delta_{t_i}$
  - $\leq f(S) + \sum_{i=1}^{k} \delta_i \quad \Rightarrow \quad \boxed{f(T) \leq f(S) + \sum_{i=1}^{k} \delta_i}$

Speeding Up Hill Climbing: Lazy Hill Climbing

Background: Submodular Functions

Hill-climbing



Add node with highest marginal gain

What do we know about optimizing submodular functions?

- A hill-climbing is near optimal ($1-1/e$ (~63%) of OPT)
- But
  - Hill-climbing algorithm is slow
    - At each iteration we need to re-evaluate marginal gains
    - It scales as $O(n\ k)$

Speeding up Hill-Climbing

- **In round i+1:** So far we picked $S_i = \{s_1,\ldots,s_i\}$
  - Now pick $s_{i+1} = \text{argmax}_u\ F(S_i \cup \{u\}) - F(S_i)$
    - maximize the "marginal benefit" $\delta_u(S_i) = F(S_i \cup \{u\}) - F(S_i)$
- **By submodularity property:**

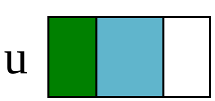
$$f(S_i \cup \{u\}) - f(S_i) \geq f(S_j \cup \{u\}) - f(S_j) \text{ for } i<j$$

- **Observation:** Submodularity implies

$$i \leq j \Rightarrow \delta_x(S_i) \geq \delta_x(S_j) \quad \text{since } S_i \subseteq S_j$$

Marginal benefits δ_x only shrink!

$\delta_u(S_i) \geq \delta_u(S_{i+1})$

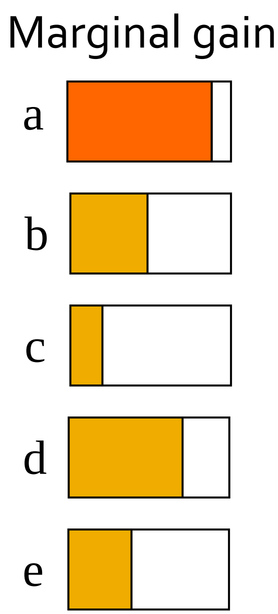


Activating node *u* in step *i* helps more than activating it at step j (j>i)

[Leskovec et al., KDD '07]

Lazy Hill Climbing

- **Idea:**
  - Use δ_i as upper-bound on δ_j (j>i)
- **Lazy hill-climbing:**
  - Keep an ordered list of marginal benefits δ_i from previous iteration
  - Re-evaluate δ_i only for top node
  - Re-sort and prune

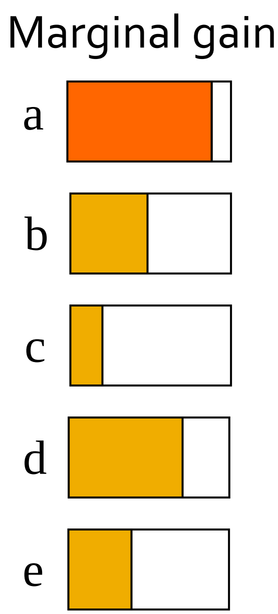


$S_1=\{a\}$

$$f(S \cup \{u\}) - f(S) \geq f(T \cup \{u\}) - f(T) \qquad S \subseteq T$$

Lazy Hill Climbing

- **Idea:**
  - Use δ_i as upper-bound on δ_j (j>i)
- **Lazy hill-climbing:**
  - Keep an ordered list of marginal benefits δ_i from previous iteration
  - Re-evaluate δ_i only for top node
  - Re-sort and prune



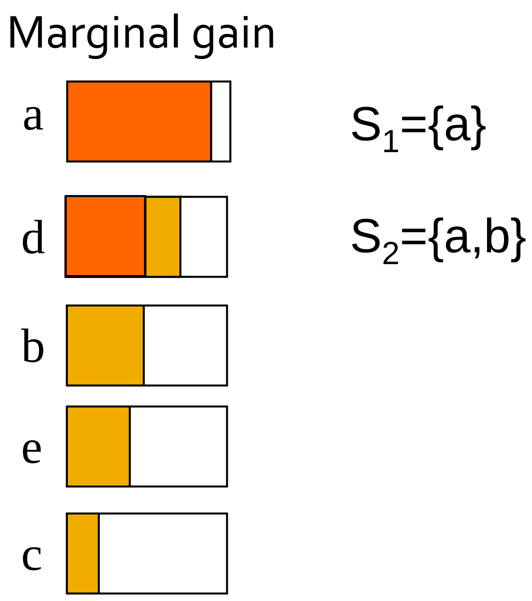
$S_1=\{a\}$

$$f(S \cup \{u\}) - f(S) \geq f(T \cup \{u\}) - f(T) \qquad S \subseteq T$$

[Leskovec et al., KDD '07]

Lazy Hill Climbing

- **Idea:**
  - Use δ_i as upper-bound on δ_j (j>i)
- **Lazy hill-climbing**:
  - Keep an ordered list of marginal benefits δ_i from previous iteration
  - Re-evaluate δ_i only for top node
  - Re-sort and prune

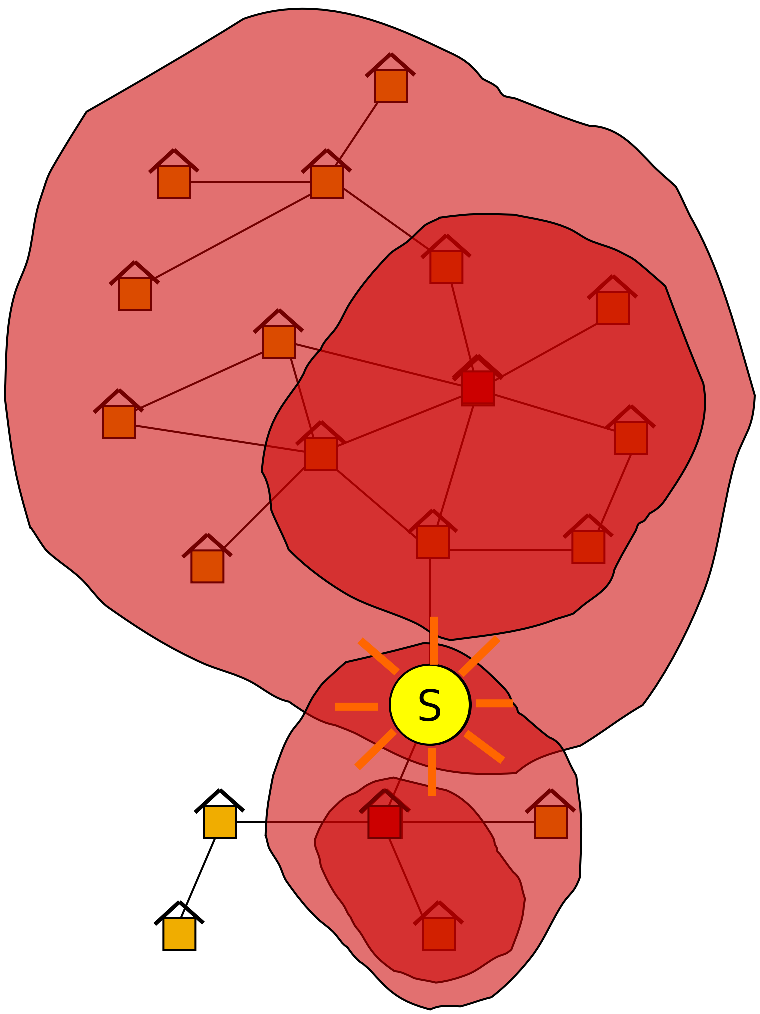


$$f(S \cup \{u\}) - f(S) \geq f(T \cup \{u\}) - f(T) \qquad S \subseteq T$$

Outbreak Detection in Networks

Problem: Water Network

- Given a real city water distribution network
- And data on how contaminants spread in the network
- Detect the contaminant as quickly as possible
- Problem posed by the *US Environmental Protection Agency*



Problem Setting

- Given a graph G(V,E)
- Data on **how outbreaks spread over the network:**
  - for each outbreak *i* we know the time *T(i,u)* when outbreak *i* contaminated node *u*
- Select a subset of nodes *A* that maximize the expected **reward**:

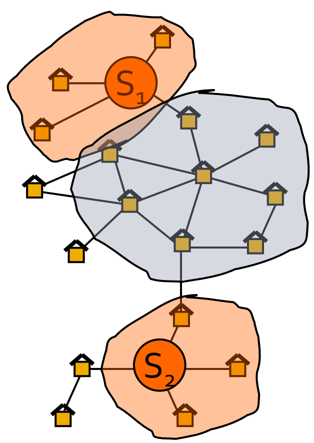
$$\max_{\mathcal{A}\subseteq\mathcal{V}} R(\mathcal{A}) \equiv \sum_i P(i) \underbrace{R_i(T(i,\mathcal{A}))}_{\text{Reward for detecting outbreak } i}$$

Reward for detecting outbreak *i*

- **Reward:** Save the most people

Structure of the Problem

- Observation: Diminishing returns

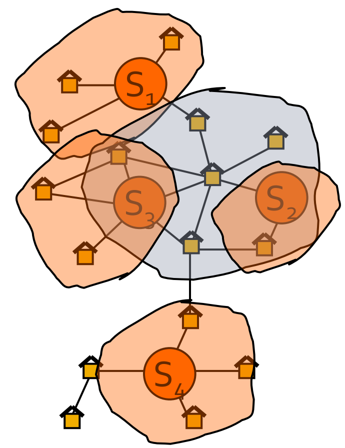


Placement A={s_1, s_2}

Adding s' helps a lot

New sensor:

s'



Placement A'={s_1, s_2, s_3, s_4}

Adding s' helps very little

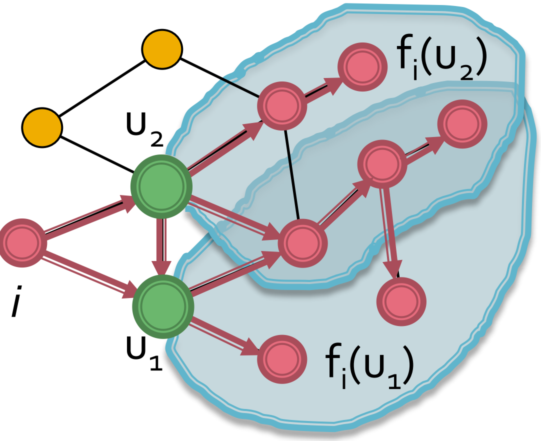
[Leskovec et al., KDD '07]

Reward Function is Submodular

- <u>Claim:</u>
  - The reward function is submodular
- Consider outbreak *i*:
  - $R_i(u_k)$ = set of nodes saved from u_k
  - $R_i(A)$ = size of union $R_i(u_k)$, $u_k \in A$

  $\Rightarrow R_i$ is submodular
- **Global optimization:**
  - $R(A) = \sum_i \text{Prob}(i)\, R_i(A)$

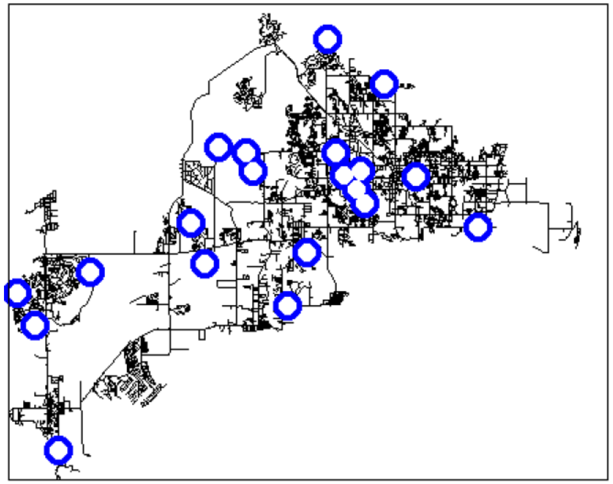
  $\Rightarrow$ R(A) is submodular



[Leskovec et al., KDD '07]

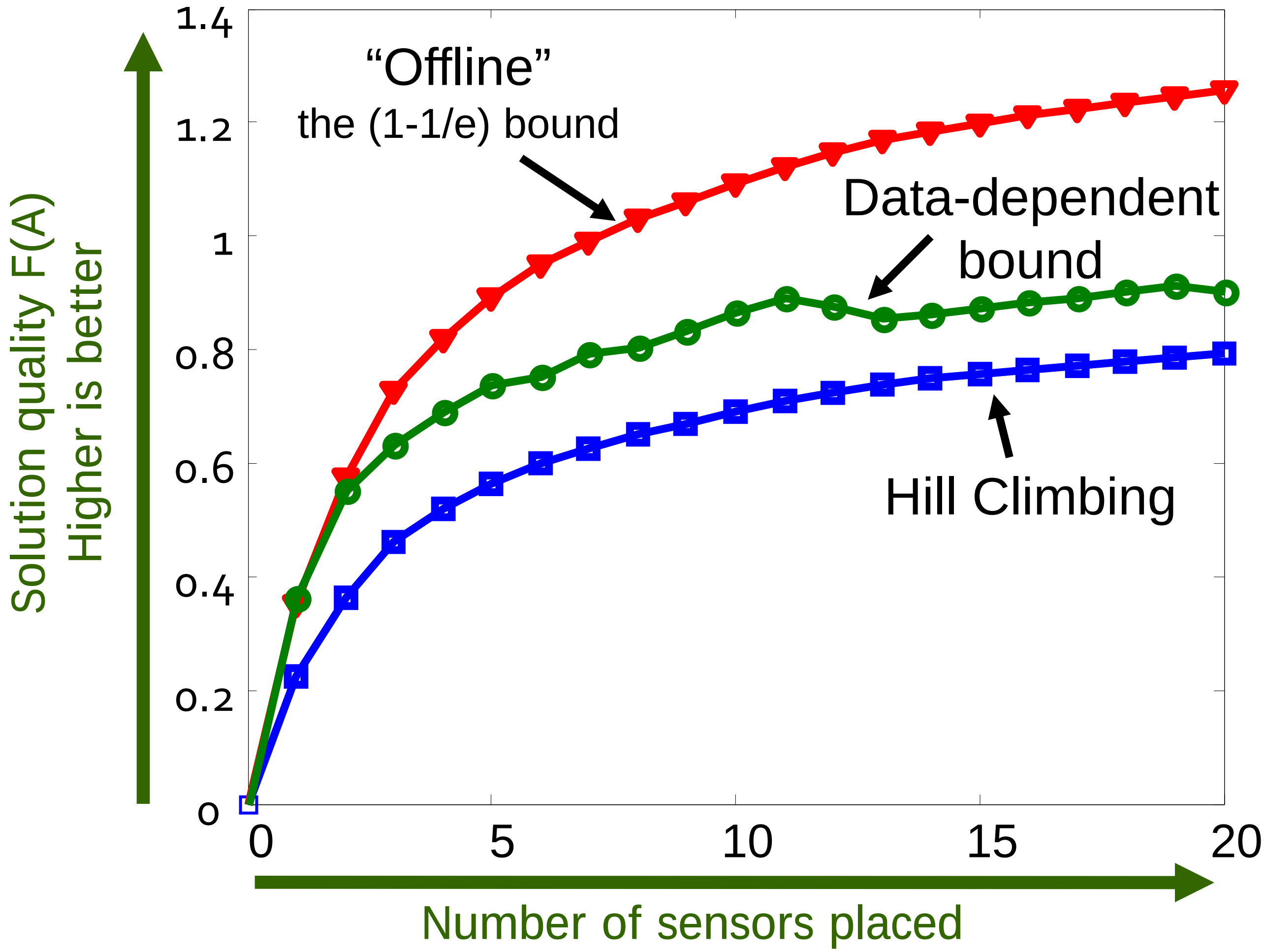
Case study: Water Network

- **Real metropolitan area network**
  - V = 21,000 nodes
  - E = 25,000 pipes



- Use a cluster of 50 machines for a month
- Simulate 3.6 million epidemic scenarios (152 GB of epidemic data)
- By exploiting sparsity we fit it into main memory (16GB)

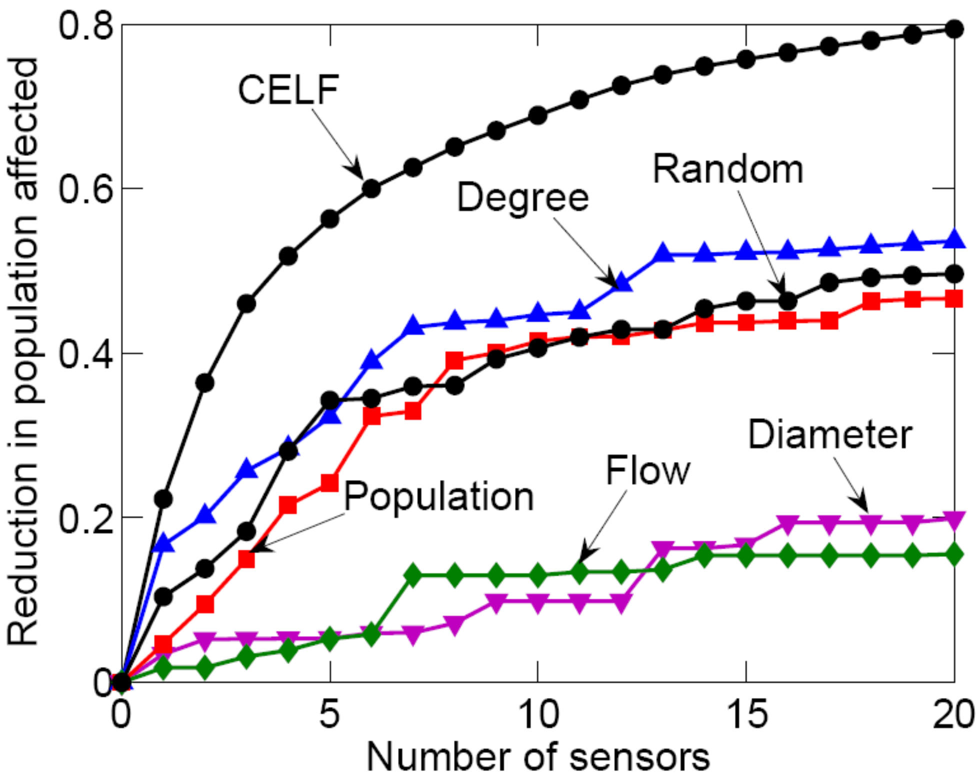
Bounds on optimal solution



Submodularity gives **data-dependent** bounds on the performance of **any** algorithm

[Leskovec et al., KDD '07]

Water: Heuristic Placement

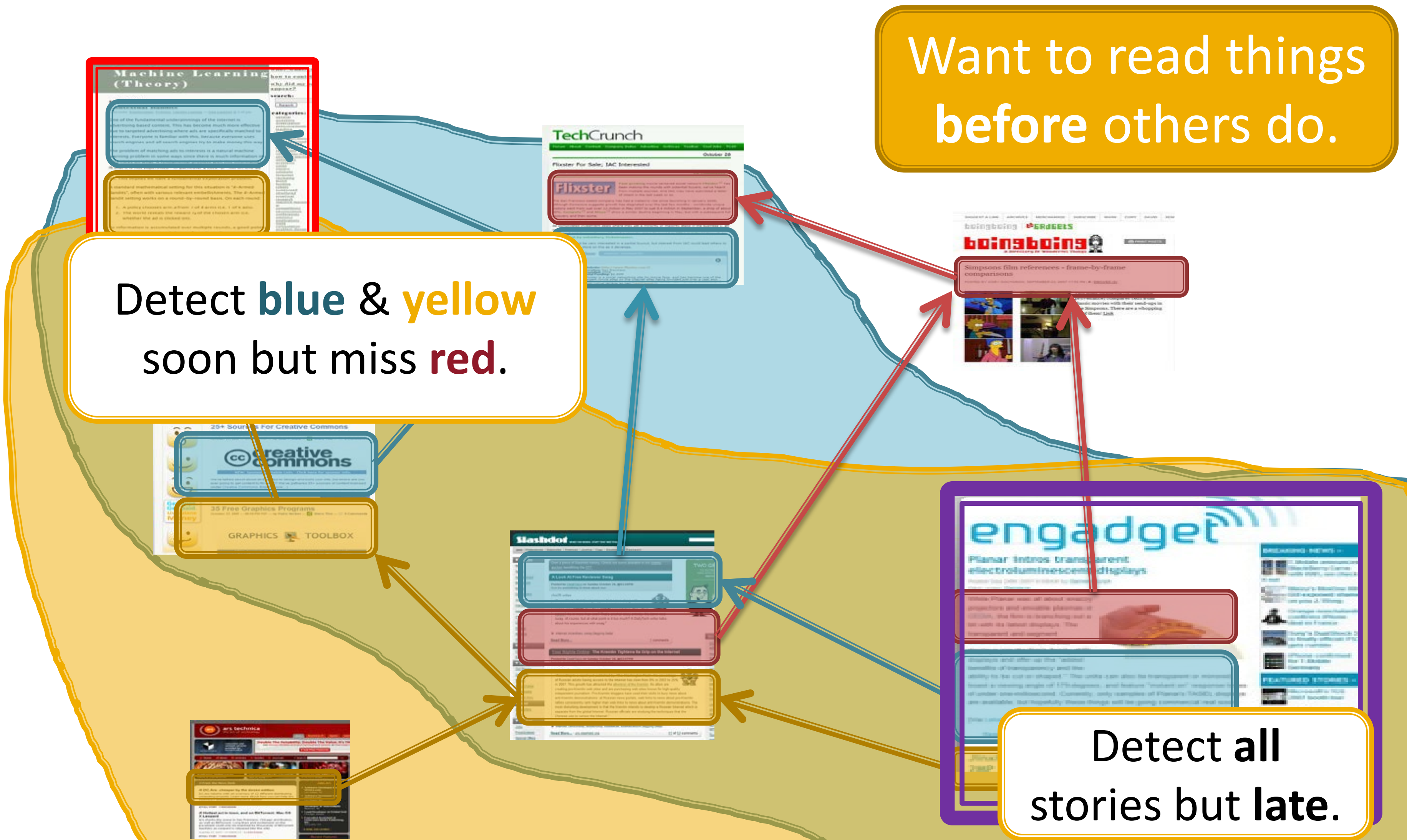


- **Placement heuristics perform much worse**

Question…

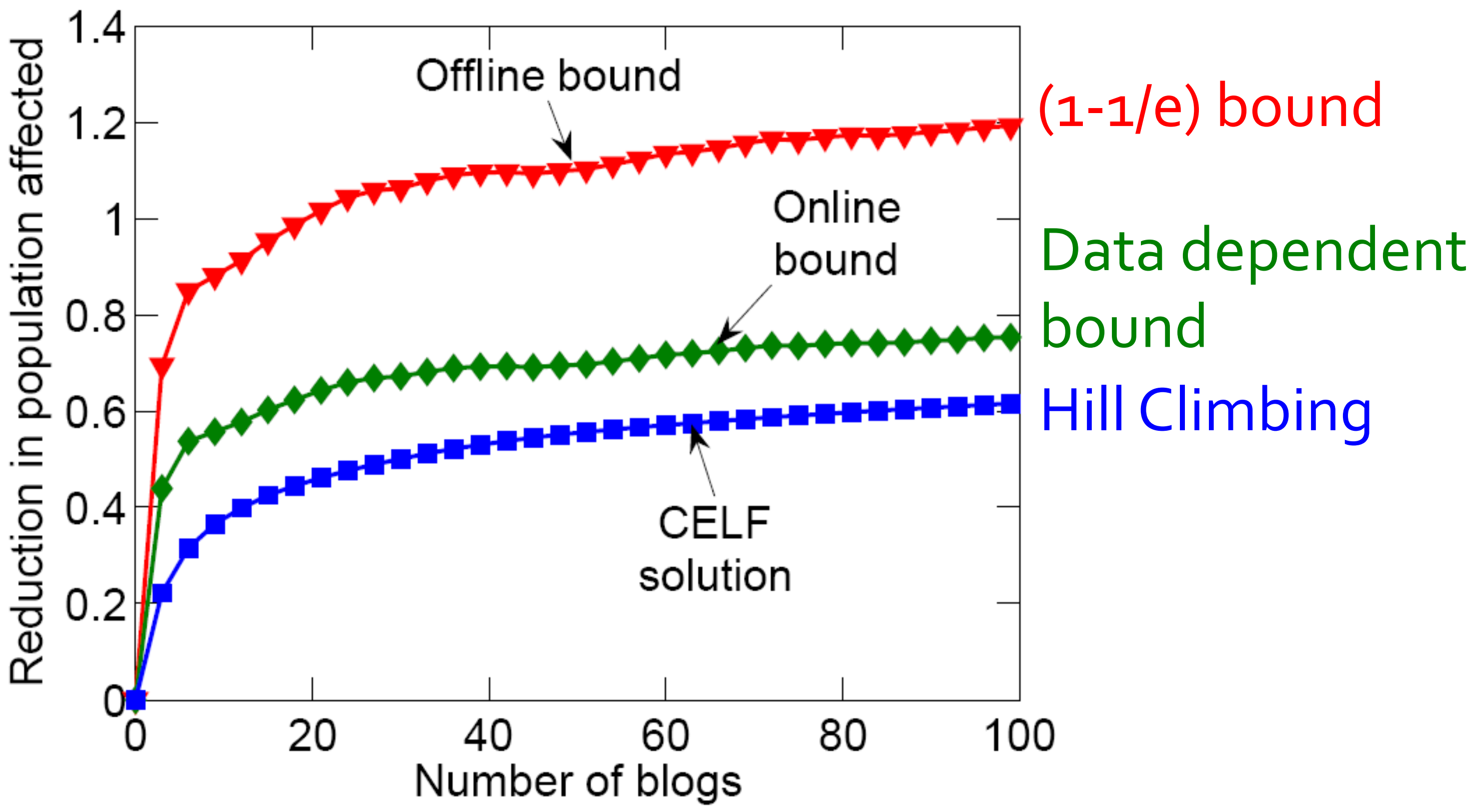
- I have 10 minutes. Which blogs should I read to be most up to date?
- Who are the most influential bloggers?

Detecting information outbreaks



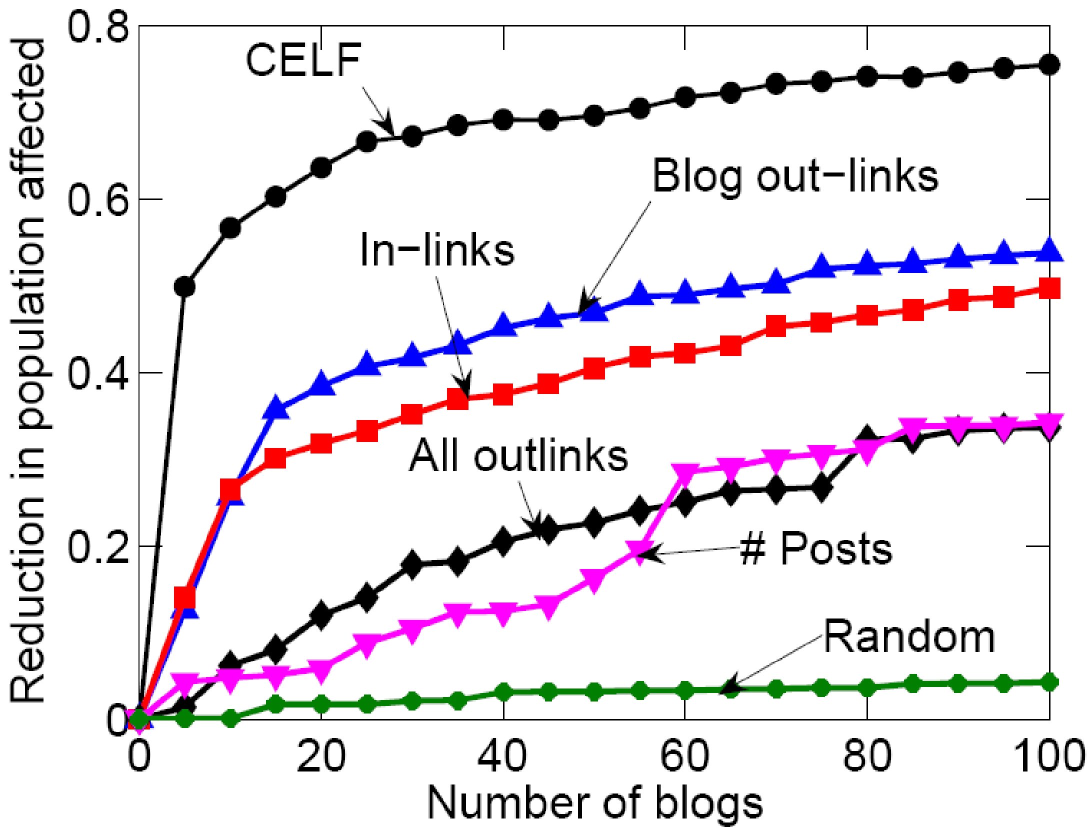
Blogs: Solution Quality

- Online bound is much tighter:
  - 13% instead of 37%



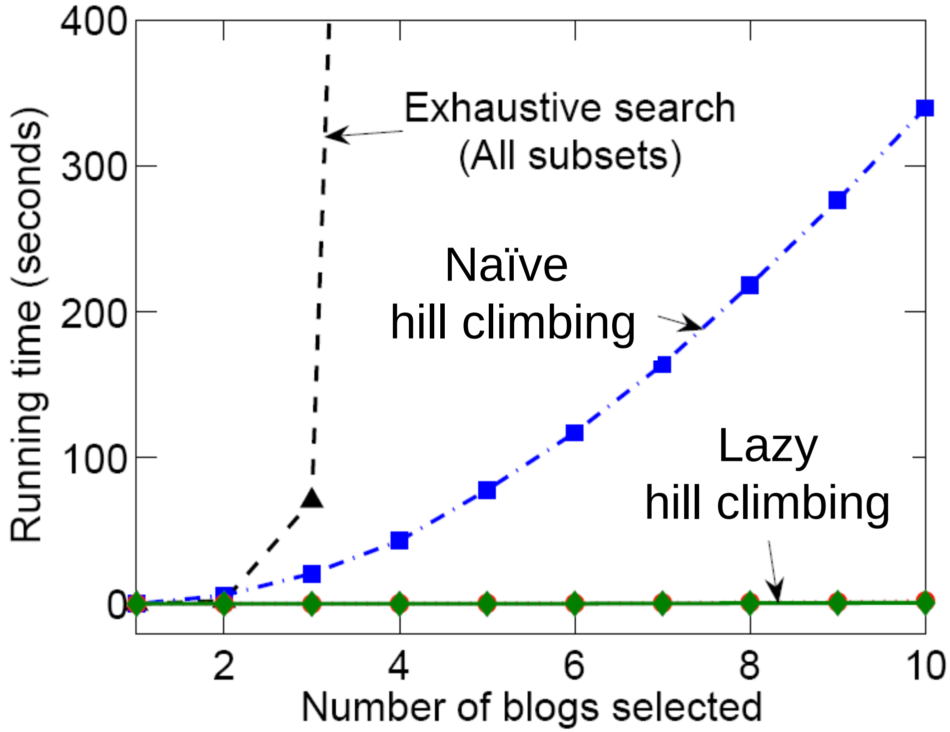
[Leskovec et al., KDD '07]

Blogs: Heuristic Selection



- **Heuristics perform much worse**

Blogs: Scalability



- Lazy evaluation runs **700** times faster than naïve Hill Climbing algorithm