Influence Maximization and Outbreak Detection

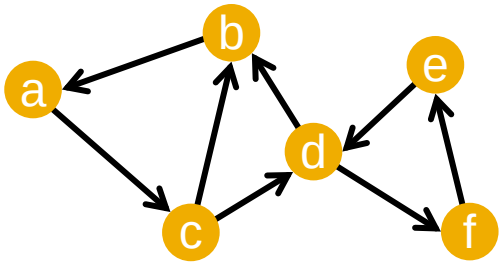
CS224W: Social and Information Network Analysis

Jure Leskovec, Stanford University

http://cs224w.stanford.edu

RECAP: Influence Maximization

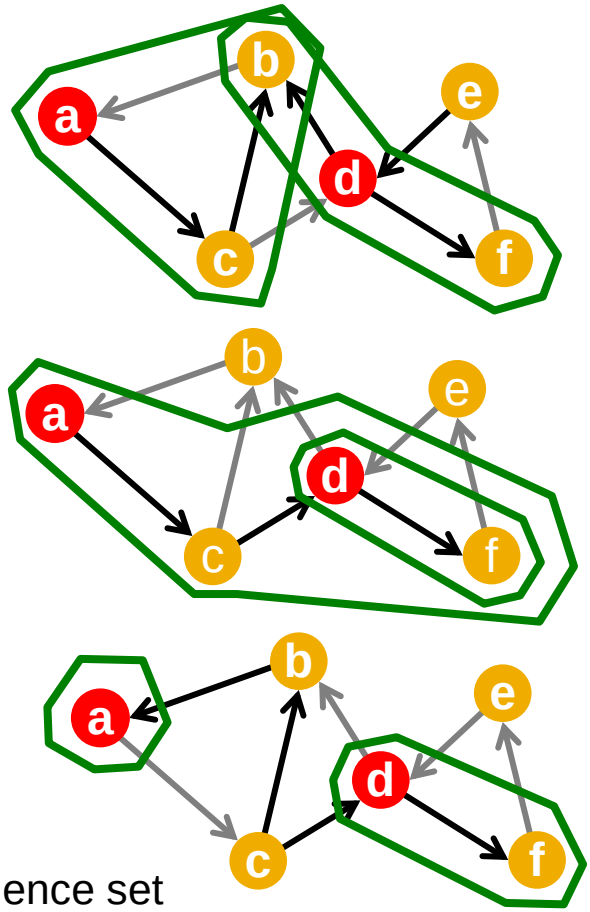
- **Find most influential set S of size *k*:** largest expected cascade size ***f(S)*** if set S is activated



Network, each edge activates with prob. p

Activate edges by coin flipping

Multiple realizations *i*:



- **Want to solve:**

$$\max_{|S|=k} f(S) = \frac{1}{|I|}\sum_{i\in I} f_i(S)$$

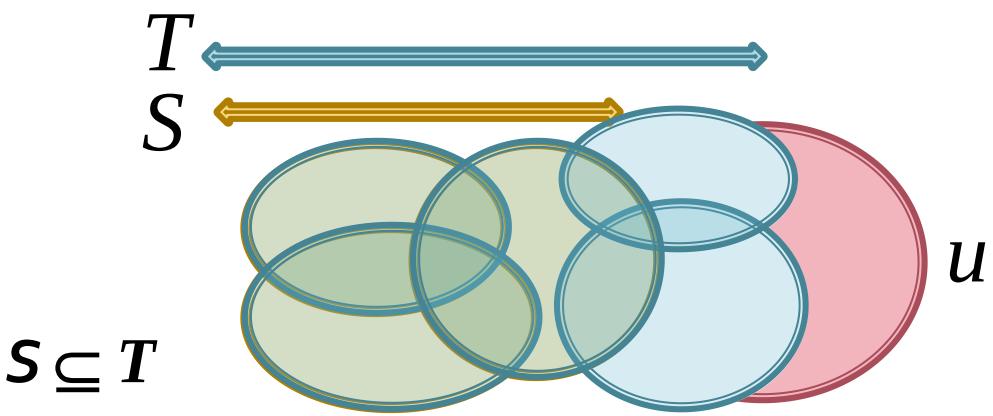
Consider S={a,d} then:
$f_1(S)=5$, $f_2(S)=4$, $f_3(S)=3$
and f(S) = 4

Approximation Guarantee

- **Algorithm:** Hill Climbing
  - At step *i* pick node *u* that: $\max_u f(S_{i-1} \cup \{u\})$
- **Thm:** Hill climbing produces a solution S where: f(S) ≥(1-1/e)*OPT
- **Claim holds for functions *f()* with 2 properties:**
  - ***f* is monotone:** ∀ **S ⊆ T:** $f(S) \leq f(T)$ and $f(\{\})=0$
  - ***f* is submodular:** $\forall S \subseteq T$ **:**
    $f(S \cup \{u\}) - f(S) \geq f(T \cup \{u\}) - f(T)$
- **Our f(S) is submodular!** Why?

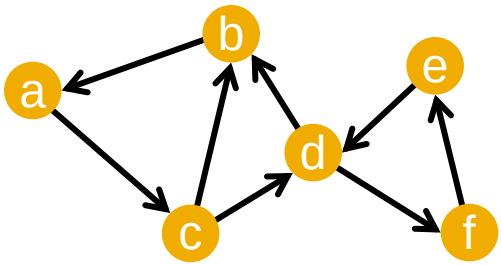
Background: Submodular Functions

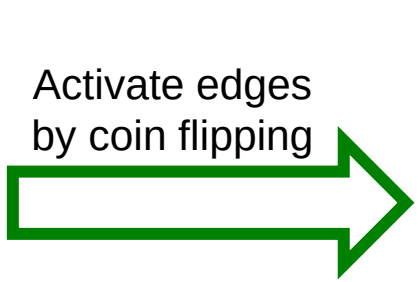
- **Submodularity:** $f(S \cup \{u\}) - f(S) \geq f(T \cup \{u\}) - f(T)$
- **Basic fact 1:**
  - If $f_1(x), \ldots, f_k(x)$ are **submodular**, and $c_1, \ldots, c_k \geq 0$ then $F(x) = \sum_i c_i \cdot f_i(x)$ is also **submodular**
- **Basic fact 2: A simple submodular function**
  - Sets $A_1, \ldots, A_m$
  - $f(S) = |\cup_{i \in S} A_i|$ **(size of the union of sets A_i, $i \in S$)**

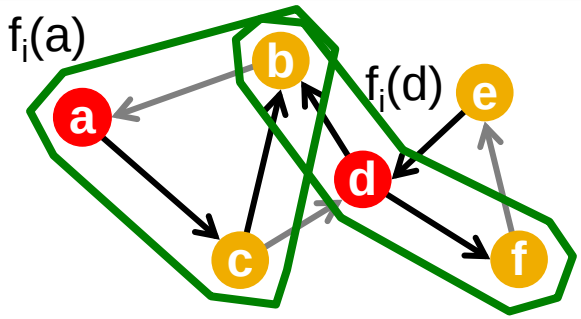


The more sets you already chose the less new area a new set *u* will cover

Our f(S) is Submodular!



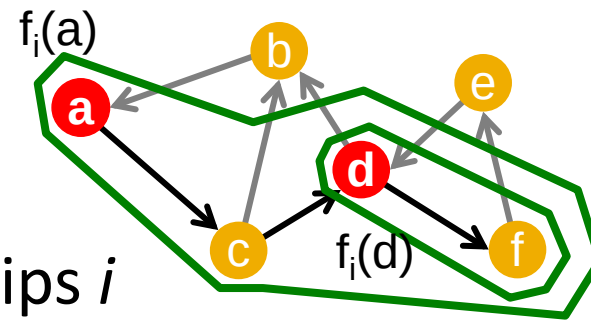


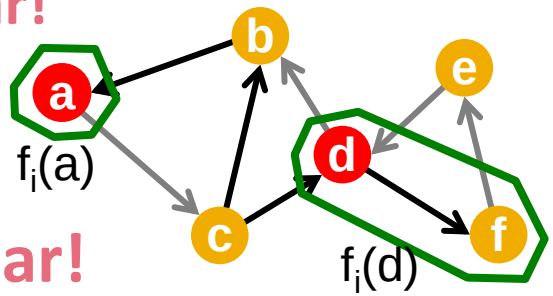


- **Fix outcome *i* of coin flips**
- $f_i(u)$ = **influence set**: set of nodes reachable from *u* on live-edge paths
- $f_i(S)$ = size of cascades from *S* on coin flips *i*
  - $f_i(S) = |\cup_{u \in S} f_i(u)| \Rightarrow f_i(S)$ is **submodular!**
    - f_i(v) are sets and f_i(S) is the size of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

$$f(S) = \frac{1}{|I|} \sum_{i \in I} f_i(S) \Rightarrow f(S) \text{ is submodular!}$$

  - 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

Claim:

If f(S) is monotone and submodular.
Hill climbing produces a solution S
where: **f(S) ≥(1-1/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,…,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) ≥ (1-1/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:

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

What is δ_i (e.i., Gain at Step i)?

Remember: $\delta_i = f(S_i) - f(S_{i-1})$

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

$T = \{t_1, \ldots t_k\}$
t_j is one choice of a next element. We greedily choose **the best one**, for a gain of δ_{i+1}. This is the "hill-climbing" assumption.

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 $f(S_k)$?

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

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 i.e., f(S) ≥ 0.63*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:

- Value of the bound depends on the input data
  - On "easy" data, hill climbing may do better than 63%

Data Dependent Bound

- Suppose S is some solution to f(S) s.t. $|S| \leq k$
  - f(S) is monotone & submodular
- Let $T = \{t_1,\ldots,t_k\}$ be the OPT solution
- 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:** $f(S) \geq f(T) - \sum_{i=1}^{k} \delta_i$
  - **Note:**
    - This is a data dependent bound (δ_u depend on input data)
    - Bound holds for **any** algorithm
      - Makes no assumption about how S is computed
    - For some inputs the bound can be "loose" (worse than 63%)

Data Dependent Bound

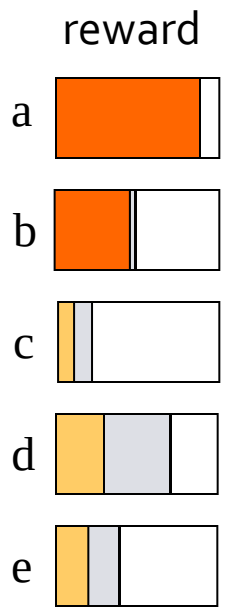
- For each u $\notin$ S let δ_u = f(S$\cup${u})-f(S)
  Order δ_u so that $\delta_1 \geq \delta_2 \geq \ldots \geq \delta_n$
  Then: $f(S) \geq f(T) - \sum_{i=1}^{k} \delta_i$
- **Proof:**
  - $f(T) \leq f(T \cup S) =$
    $f(S) + \sum_{i=1}^{k} [f(S \cup \{t_1 \ldots t_i\}) - f(S \cup \{t_1 \ldots t_{i-1}\})]$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

Instead of adding $t_i \in T$ (of benefit δ_{t_i}),
we add the best possible element (δ_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)*

[Leskovec et al., KDD '07]

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:** By submodularity : For some u
  $\delta_u(S_i) \geq \delta_u(S_j)$ for $i \leq j$ since $S_i \subseteq S_j$

Marginal benefits δ_x only shrink!

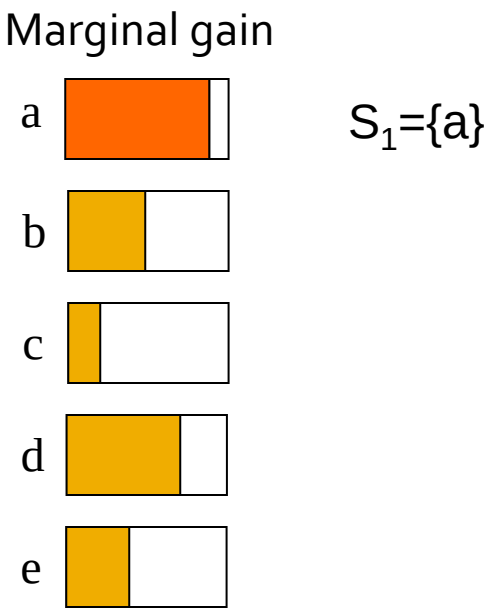
$\delta_u(S_i) \geq \delta_u(S_j)$

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

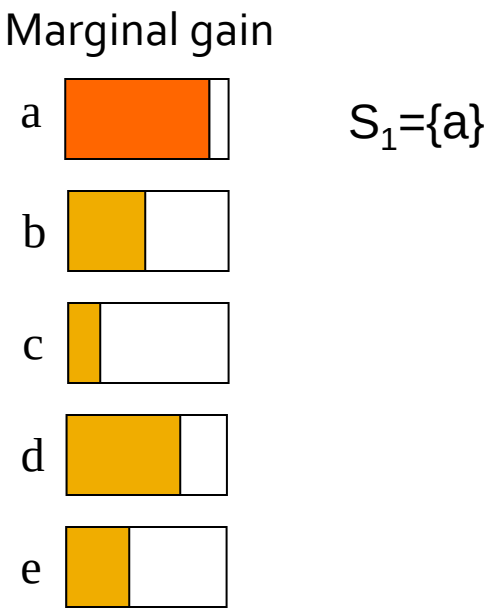


$$f(S \cup \{u\}) - f(S) \geq f(T \cup \{u\}) - f(T) \qquad S \subseteq T$$

[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

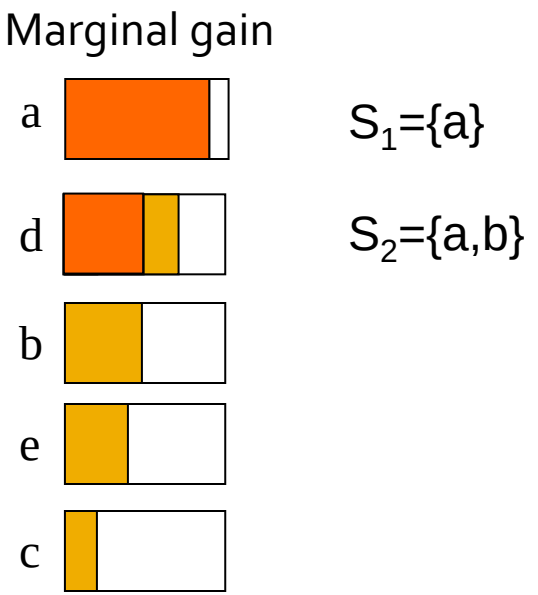


$$f(S \cup \{u\}) - f(S) \geq f(T \cup \{u\}) - f(T) \qquad S \subseteq T$$

[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



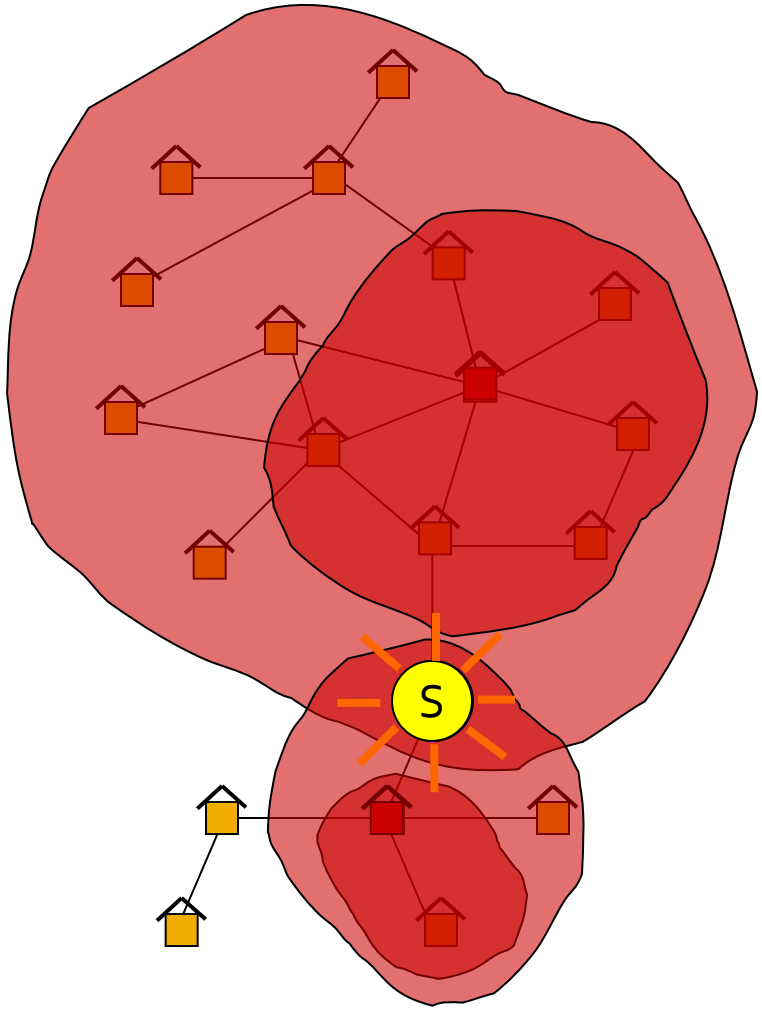
$$f(S \cup \{u\}) - f(S) \geq f(T \cup \{u\}) - f(T) \qquad S \subseteq T$$

Outbreak Detection in Networks

[Leskovec et al., KDD '07]

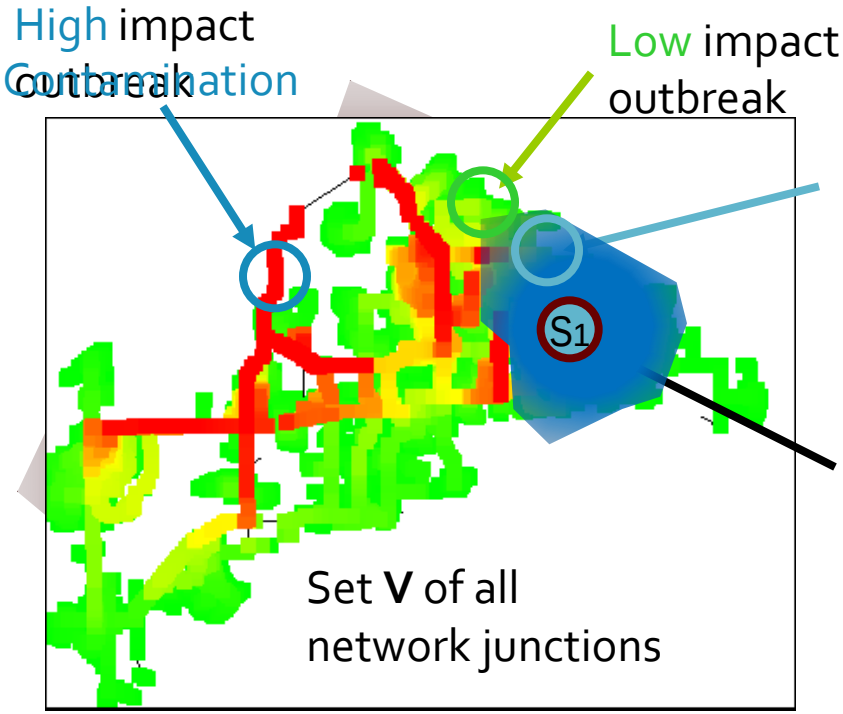
Problem: Water Network

- 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*

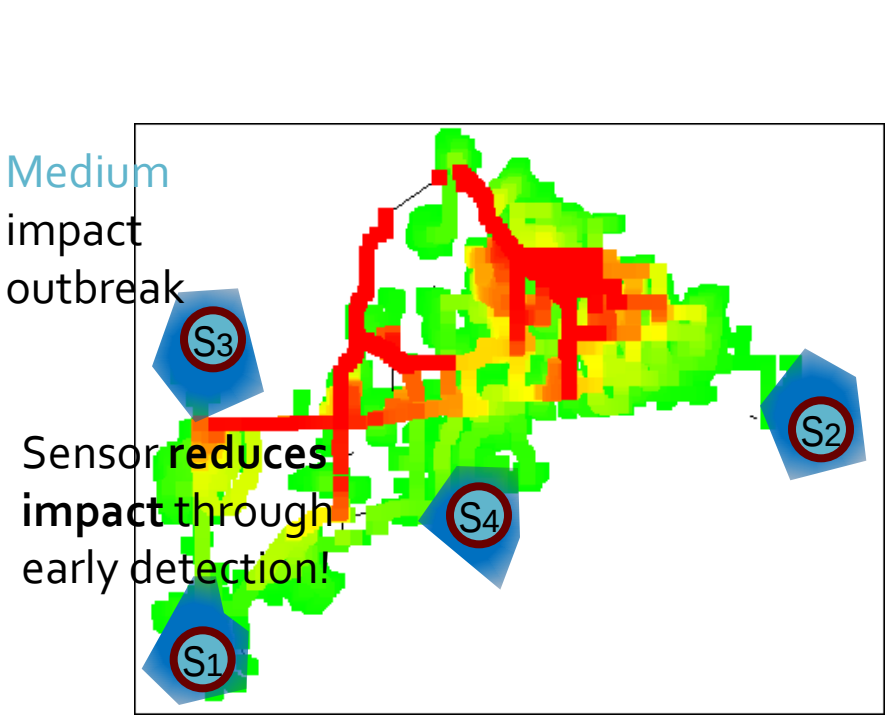


Water Network: Utility

- **Utility of placing sensors**
  - Water flow dynamics, demands of households, …
- For each subset $A \subseteq V$ compute utility F(A)



High sensing quality F(A) = 0.9



Low sensing quality F(A)=0.01

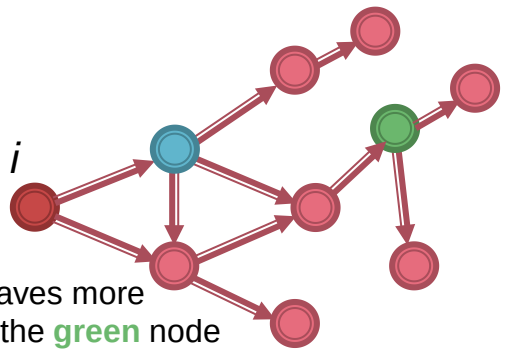
[Leskovec et al., KDD '07]

Problem Setting

- Given a graph G(V,E)
- And the 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*
- **Goal:** Select a subset of nodes *A* that maximize the expected **reward**:

$$\max_{S \subseteq V} f(S) = \sum_i \underbrace{P(i) f_i(S)}_{\text{Expected reward for detecting outbreak } i}$$

- **Reward:** Raise the alarm and save the most people

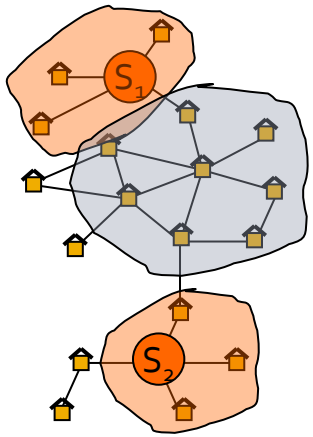


Monitoring **blue** node saves more people than monitoring the **green** node

[Leskovec et al., KDD '07]

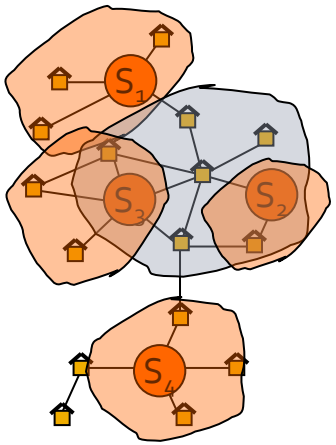
Structure of the Problem

- **Observation: Diminishing returns**



Placement $S=\{s_1, s_2\}$

New sensor:



Placement $S'=\{s_1, s_2, s_3, s_4\}$

Adding s' helps a lot

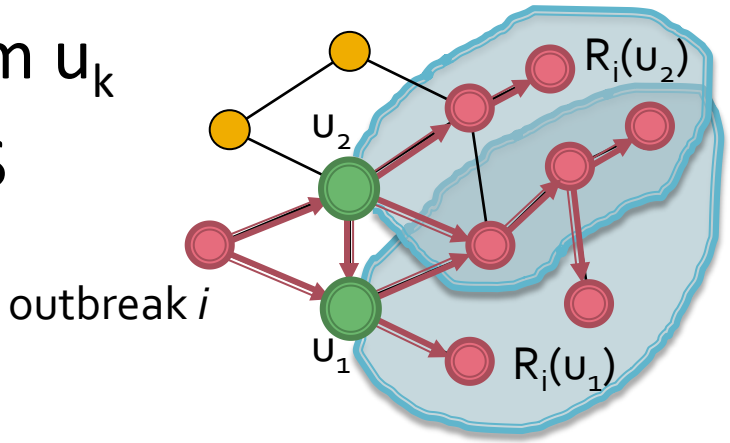
Adding s' helps very little

[Leskovec et al., KDD '07]

Reward Function is Submodular

- **Claim:**
  - The reward function is submodular
- Consider outbreak *i*:
  - $f_i(u_k)$ = set of nodes saved from u_k
  - $f_i(S)$ = size of union $f_i(u_k)$, $u_k \in S$

  $\Rightarrow f_i$ is **submodular**



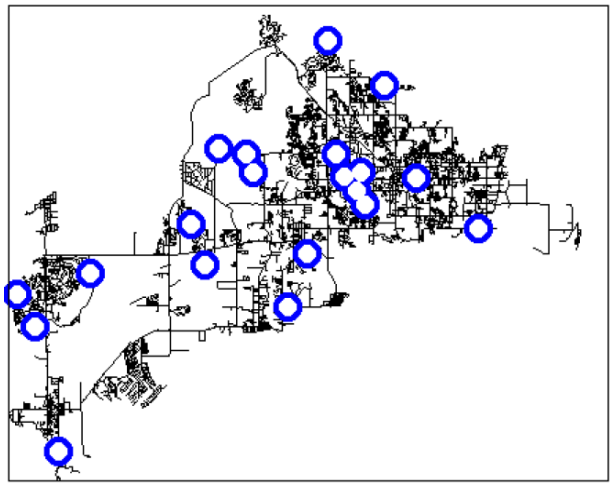
- **Global optimization:**
  - $f(S) = \sum_i P(i)\, f_i(S)$

  $\Rightarrow$ f(S) is submodular

[Leskovec et al., KDD '07]

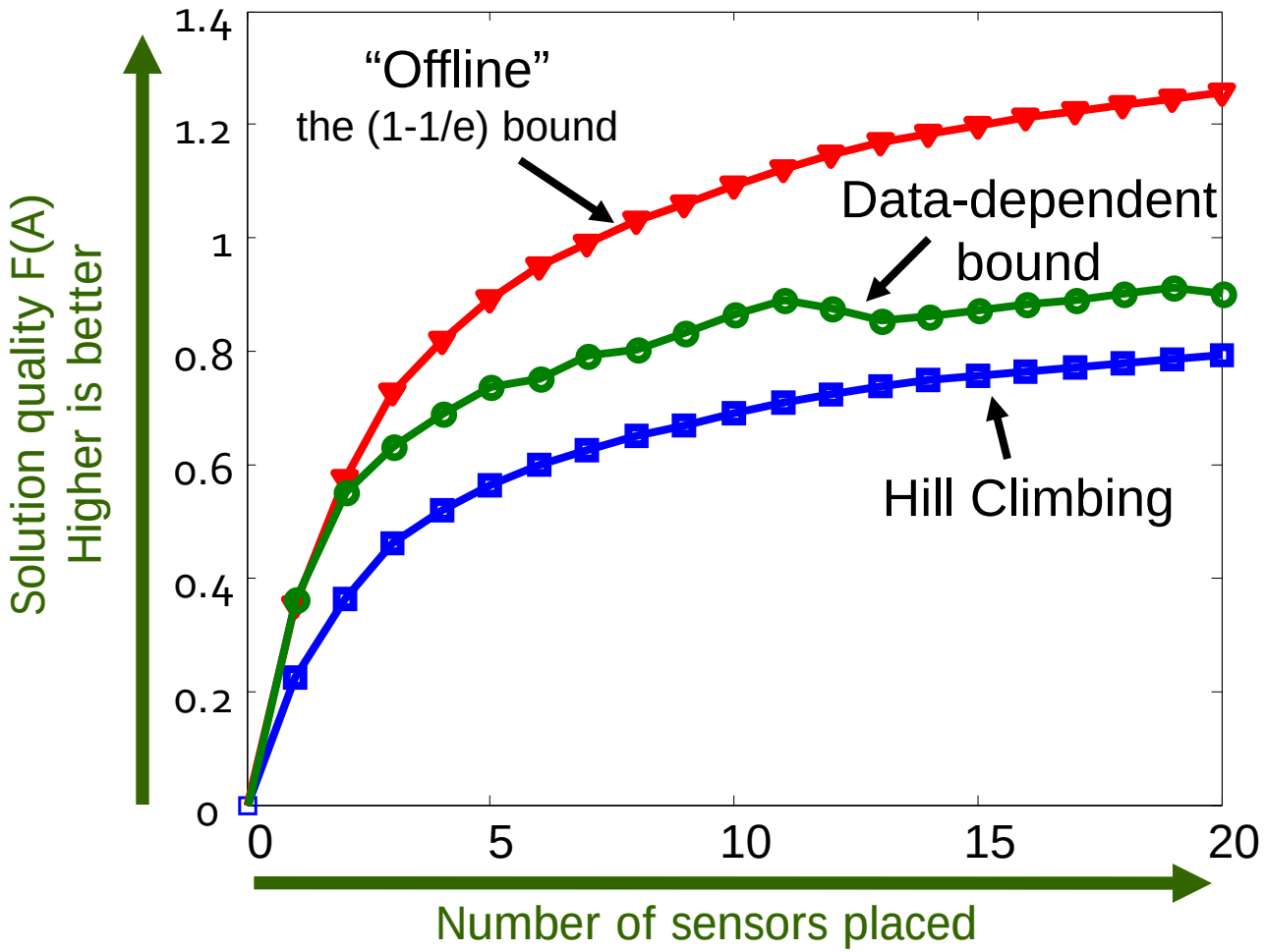
Case study: Water Network

- **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)

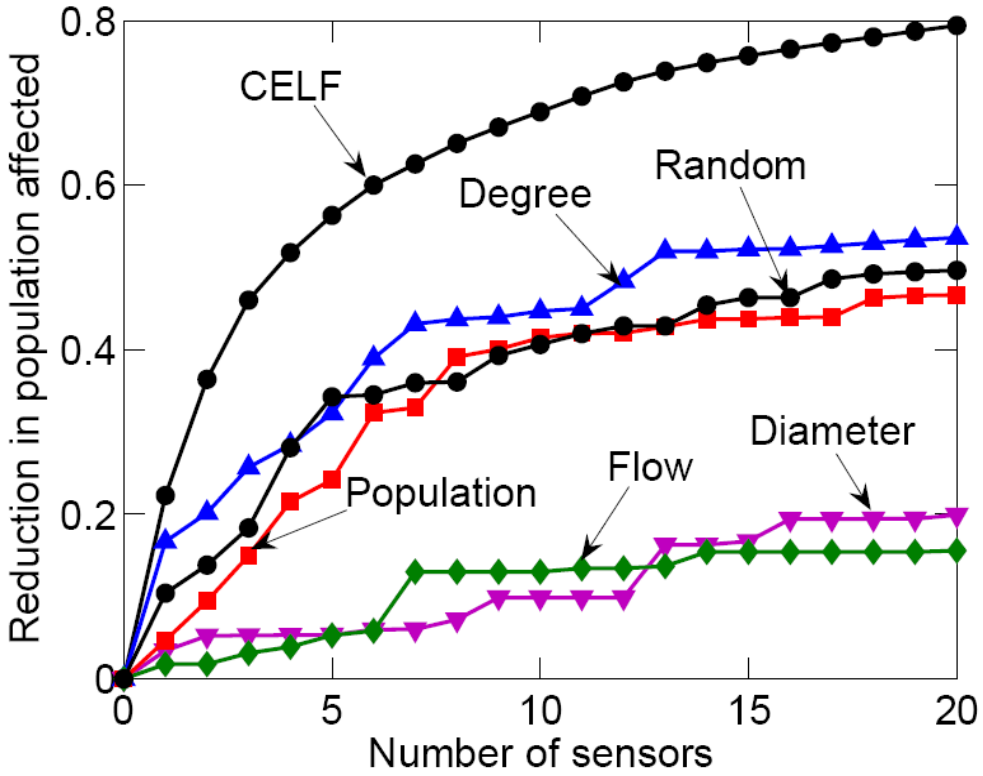
Bounds on optimal solution



Data-dependent bound is much tighter
(gives more accurate estimate of alg. performance)

[Leskovec et al., KDD '07]

Water: Heuristic Placement



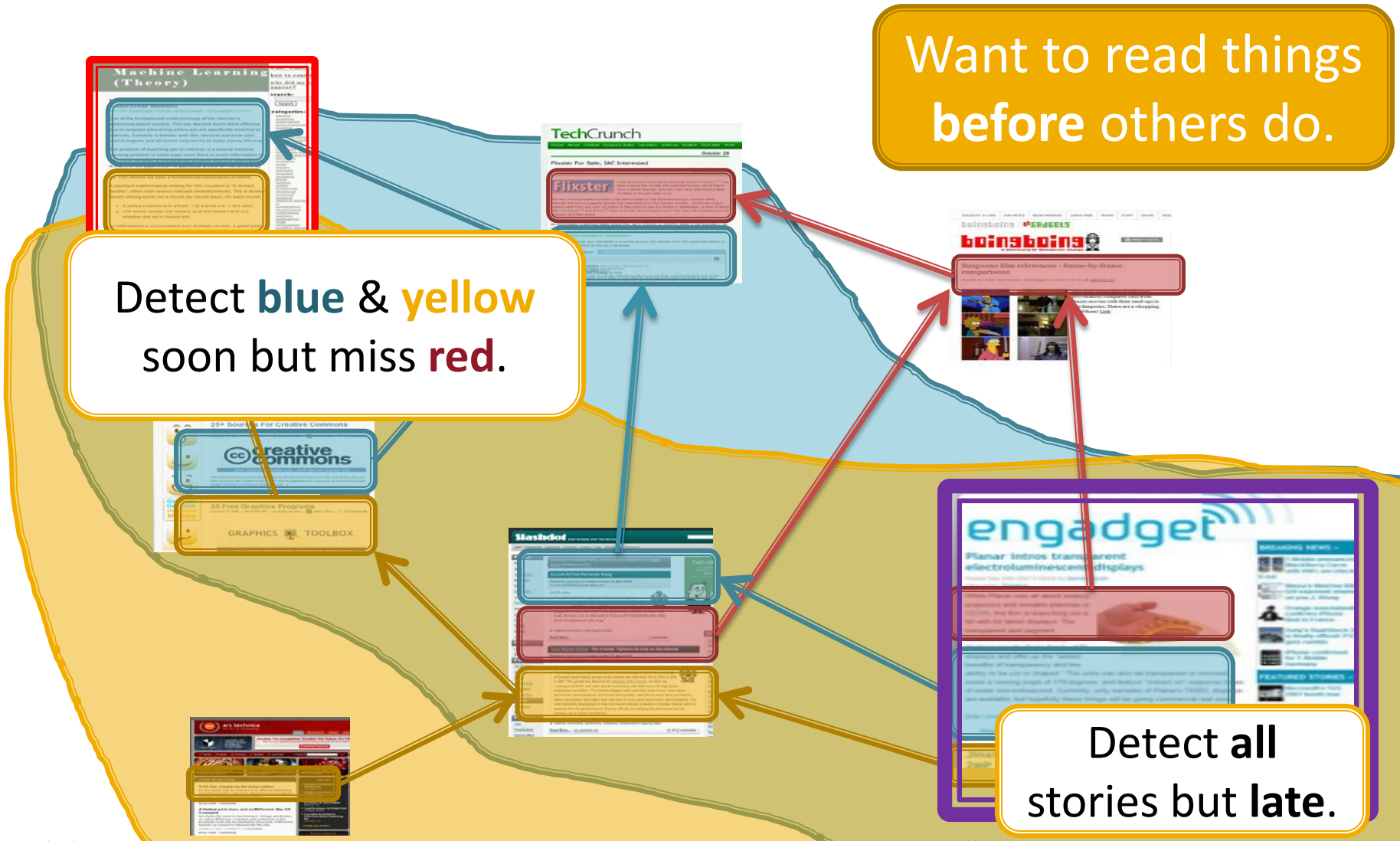
- **Placement heuristics perform much worse**

Question…

= I have 10 minutes. Which blogs should I read to be most up to date?

= Who are the most influential bloggers?

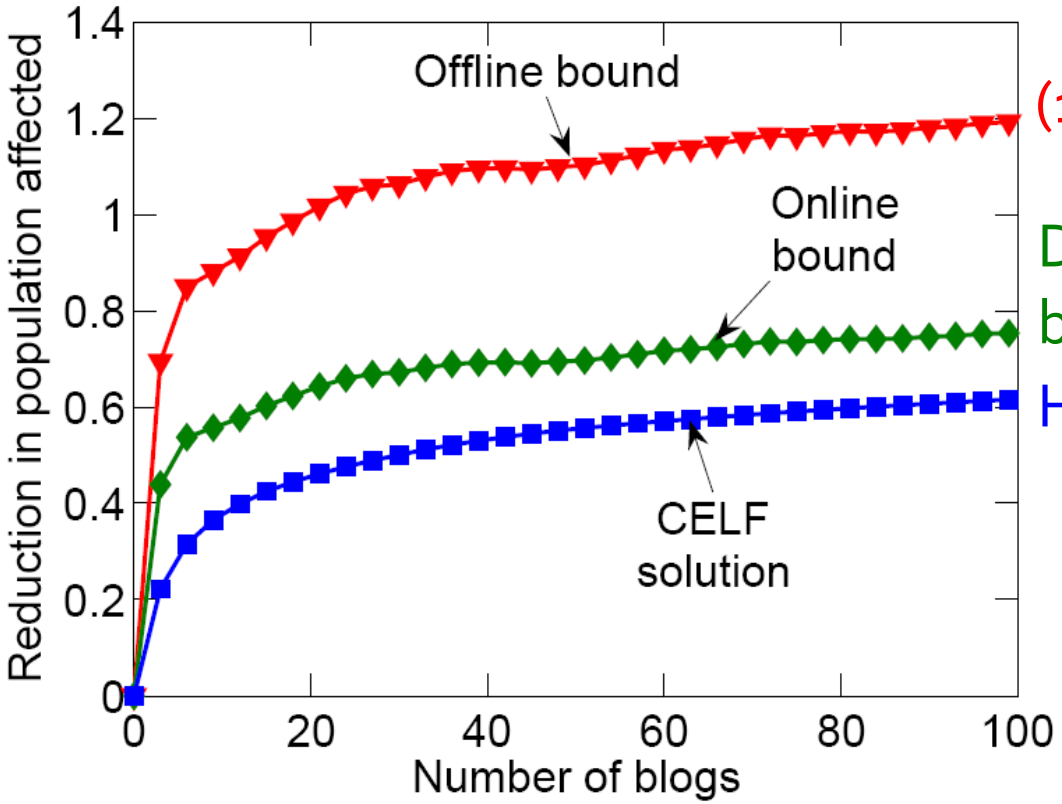
Detecting information outbreaks



[Leskovec et al., KDD '07]

Blogs: Solution Quality

- **Online bound is much tighter:**
  - 87% instead of 63%



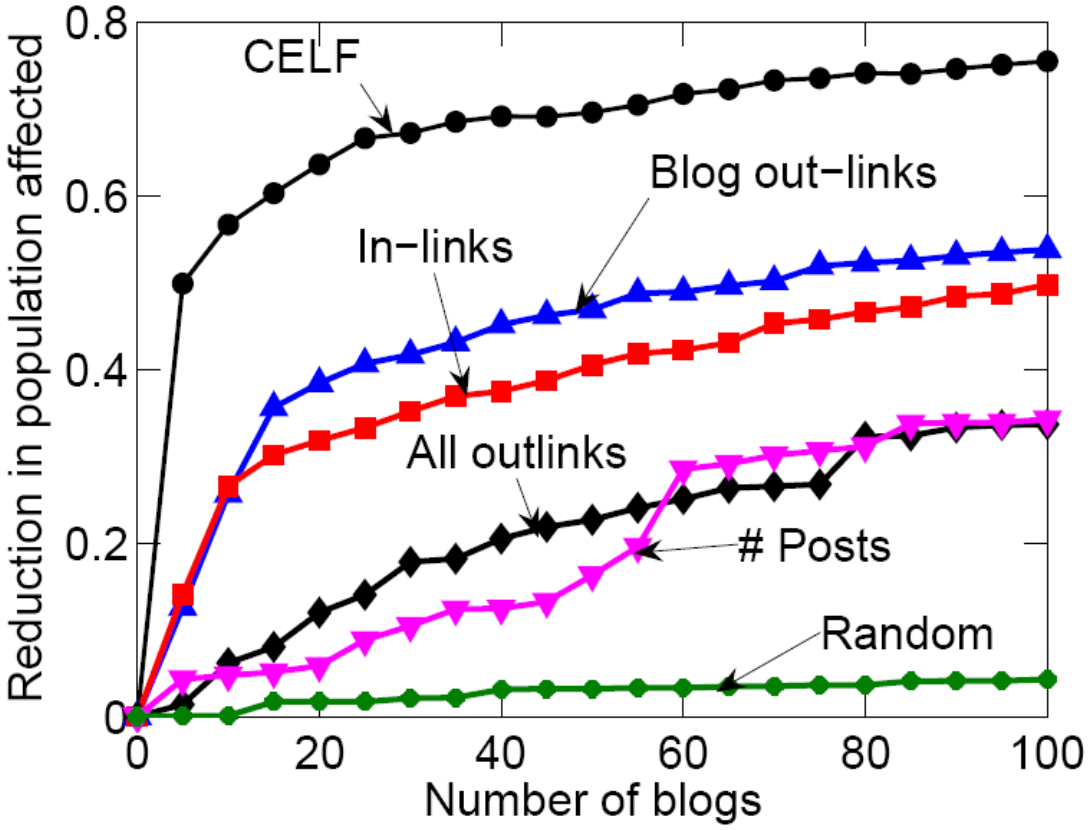
(1-1/e) bound

Data dependent bound

Hill Climbing

[Leskovec et al., KDD '07]

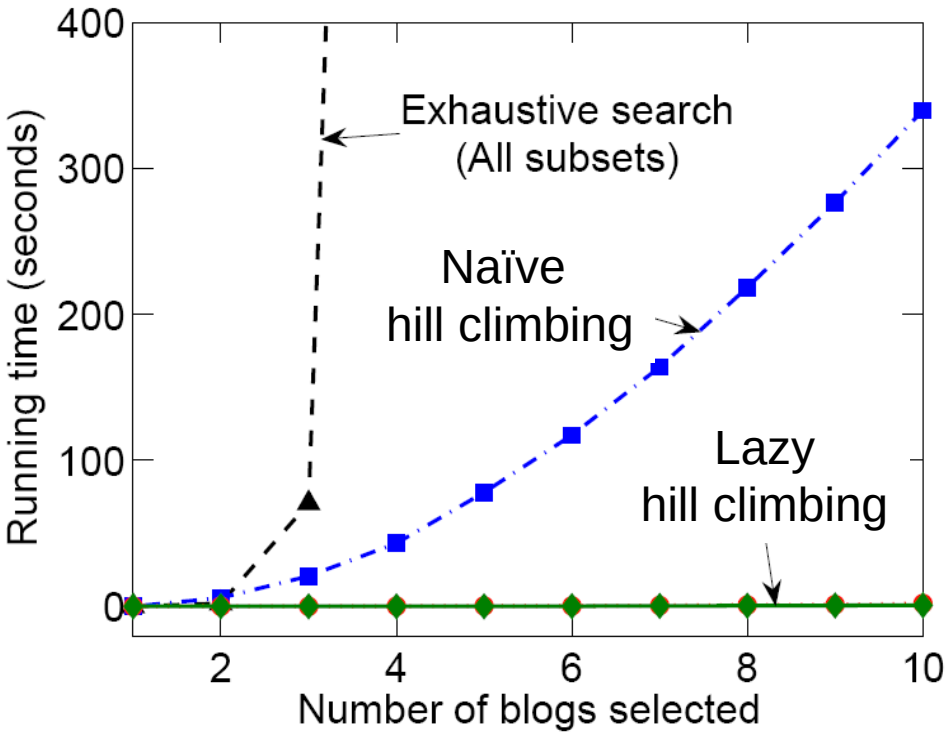
Blogs: Heuristic Selection



- **Heuristics perform much worse**

[Leskovec et al., KDD '07]

Blogs: Scalability



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