

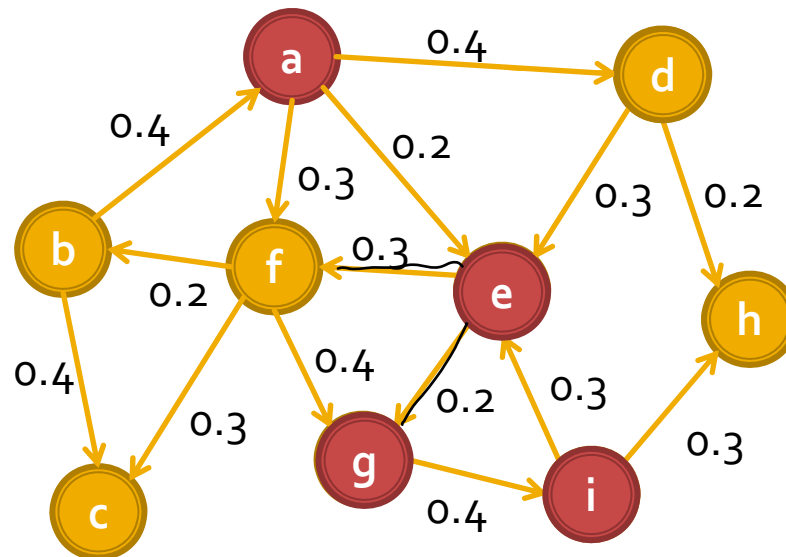
Influence Maximization in Networks

CS 322: (Social and Information) Network Analysis
Jure Leskovec
Stanford University



Independent Contagion Model

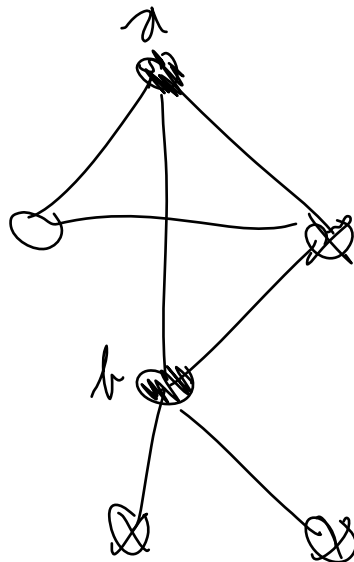
- Initially some nodes S are active
- Each edge (a,b) has probability (weight) p_{ab}



- Node a becomes active:
 - activates node b with prob. p_{ab}
- Activations spread through the network

Most Influential Subset of Nodes

- If S is initial active set, let $f(S)$ denote expected size of final active set



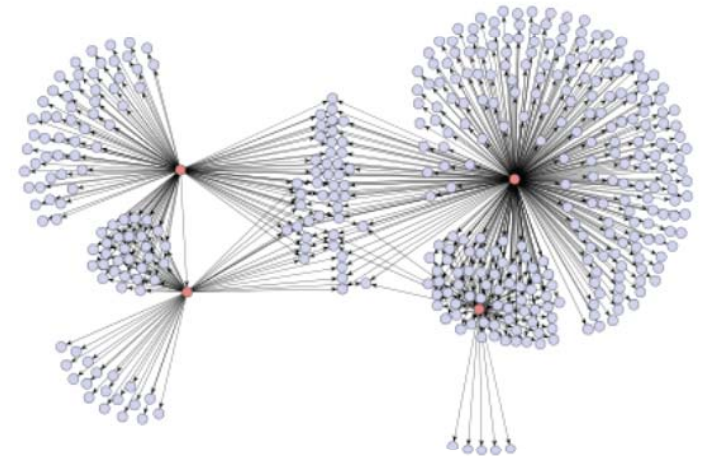
$$S = \{v_1, v_2\}$$

$$f(S) = 5$$

- S is more influential if $f(S)$ is larger

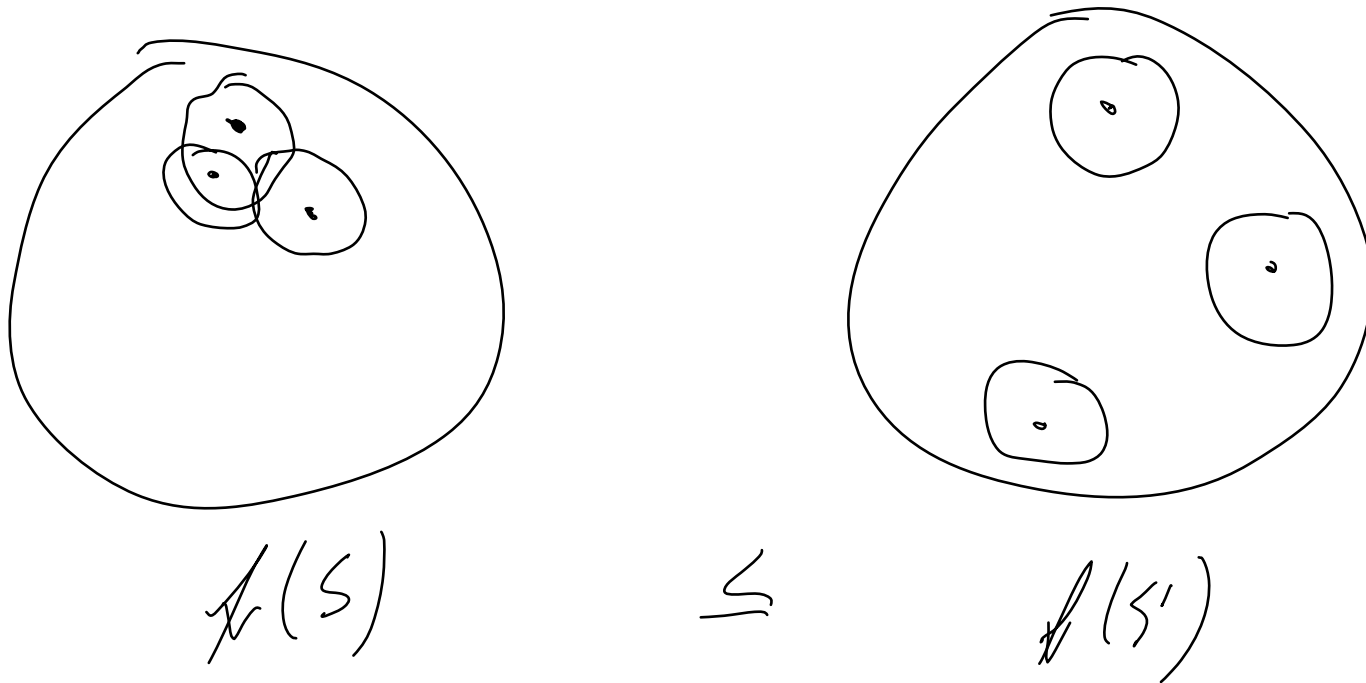
Finding influencers

- Blogs – information epidemics
 - Which are the influential/infectious blogs?
- Viral marketing
 - Who are the trendsetters?
 - Influential people?
- Disease spreading
 - Where to place monitoring stations to detect epidemics?



Most Influential Subset of Nodes

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



Most Influential Subset of Nodes

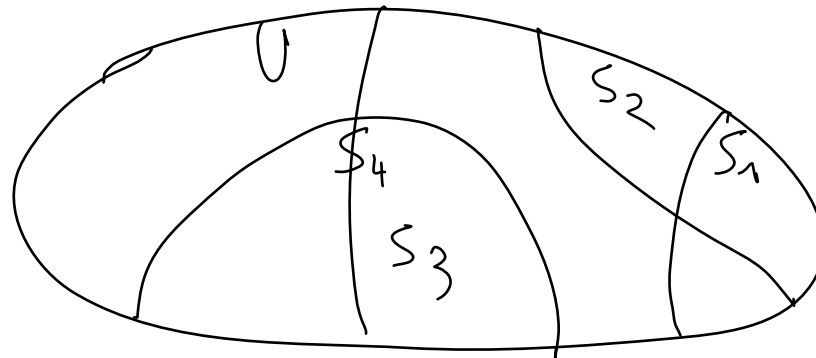
- Optimization problem:

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

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

Set cover

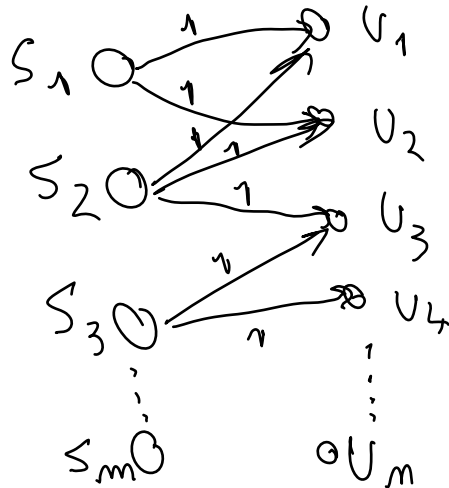
- Set cover:
 - Given $U = \{u_1, \dots, u_n\}$ and sets $S_1, \dots, S_m \subseteq U$
 - Are there k sets among S_1, \dots, S_m such that their union is U ?



- Goal:
Encode set cover as an instance of $\max_{S \text{ of size } k} f(S)$

Reduction

- Given a set cover instance with sets S_1, \dots, S_m
- Build a graph:



-- Create edge (S_i, u)
 $\forall S_i \forall u \in S_i$.
-- Directed edge from all sets to 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

Approximation algorithm

- Bad news:
 - Influence maximization is NP-hard
- Good news:
 - There exists an approximation algorithm!
- Consider:
 - Greedy hill-climbing to find a good set S

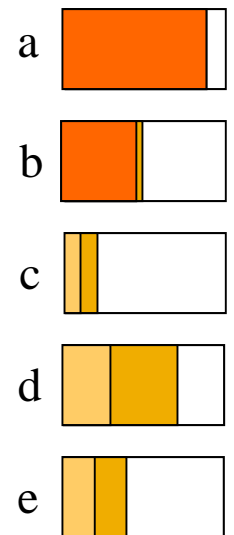
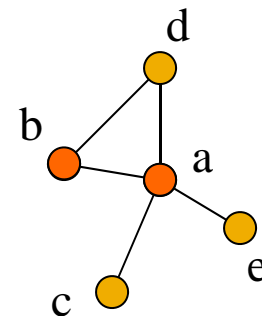
Hill climbing

- Start with $S_0 = \{\}$
- For $i = 1 \dots k$
 - Choose node v that $\max f(S_{i-1} \cup \{v\})$
 - Let $S_i = S_{i-1} \cup \{v\}$
- What is the runtime?
 - Each step just runs n time steps for each node v

$$S_0 = \{\}$$

$$S_1 = \{a\}$$

$$S_2 = \{a, b\} \quad f(S_{i-1} \cup \{v\})$$



Approximation guarantee

- Hill climbing produces a solution S where $f(S) \geq (1-1/e)$ of optimal value (~63%)

[Hemhauser, Fisher, Wolsey '78, Kempe, Kleinberg, Tardos '03]

- Claim holds for functions f with 2 properties:

- f is monotone:

if $S \subseteq T$ then $f(S) \leq f(T)$ and $f(\{\})=0$

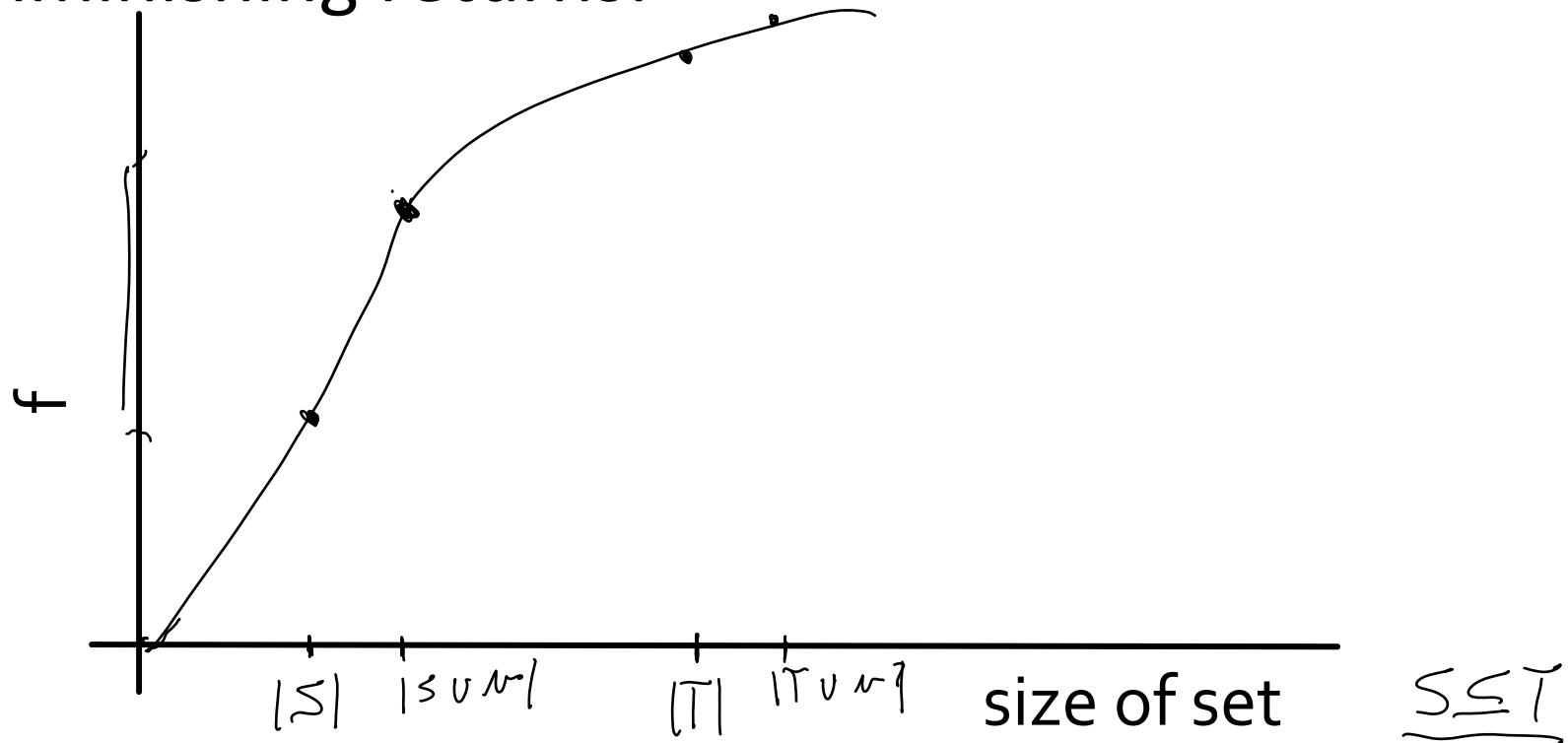
- f is submodular:

adding element to a set gives less improvement than adding to one of subsets

$$S \subseteq T \quad \forall v \in T \\ f(S \cup \{v\}) - f(S) \geq f(T \cup \{v\}) - f(T)$$

Submodularity– diminishing returns

- Diminishing returns:



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

Plan

- Show 2 things:
 - 1) Our $f(S)$ is submodular
 - 2) Hill climbing works well for monotone submodular functions

Proof for hill climbing

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

Basic fact

- Claim: $f(A \cup B) - f(A) \leq \sum_j^k f(A \cup \{b_j\}) - f(A)$

- where: $B = \{b_1, \dots, b_k\}$ and f is submodular,

- Proof: Let $B_i = \{b_1, \dots, b_i\}$
- $$\begin{aligned}
 f(A \cup B) - f(A) &= \sum_{i=1}^k f(A \cup B_i) - f(A \cup B_{i-1}) \\
 &= \sum_{i=1}^k f(A \cup B_{i-1} \cup \{b_i\}) - f(A \cup B_{i-1}) \\
 &\leq \sum_{i=1}^k f(A \cup \{b_i\}) - f(A)
 \end{aligned}$$
- $f(A \cup B_1) - f(A) + f(A \cup B_2) - f(A \cup B_1) + \dots + f(A \cup B_3) - f(A \cup B_2)$

What is δ_i ?

- $f(T) \leq f(S_i \cup T)$

$$= \underbrace{f(S_i \cup T) - f(S_i)} + f(S_i)$$

$$\leq \sum_j^k \underbrace{[f(S_i \cup \{t_j\}) - f(S_i)]} + f(S_i)$$

$$\leq \sum_j^k \delta_{i+1} + f(S_i) = k \cdot \delta_{i+1} + f(S_i)$$

Thus: $f(T) \leq f(S_i) + k \delta_{i+1} \Rightarrow \delta_{i+1} \geq \frac{1}{k} (f(T) - f(S_i))$

$$T = \{t_1 \dots t_k\}$$

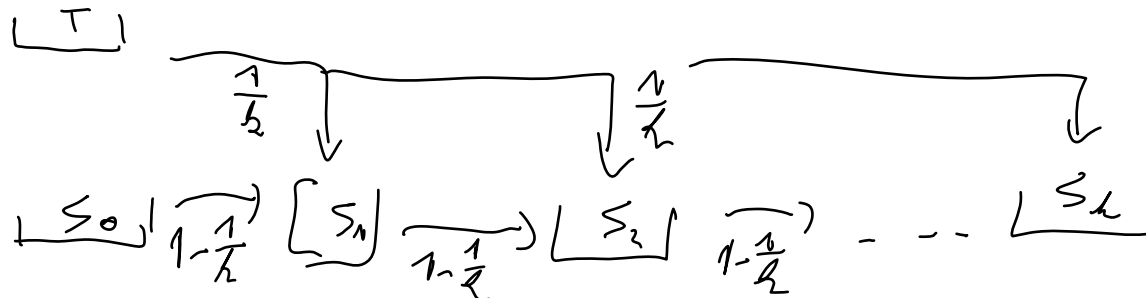
$$\delta_{i+1}$$

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

- We know: $\delta_{i+1} \geq 1/k (f(T) - f(S_i))$

- What is $f(S_{i+1}) = f(S_i) + \delta_{i+1} \geq f(S_i) + \frac{1}{k} (f(T) - f(S_i))$
 $= \left(1 - \frac{1}{k}\right) f(S_i) + \frac{1}{k} f(T)$

- 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:

- $i=0$: $f(S_0) = f(\emptyset) = 0$

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

What is $f(S_k)$?

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

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

□

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:

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

□

$$\left(1 - \frac{1}{k}\right)^k \leq \frac{1}{e}$$

$f(S)$ is submodular

- Next, we must show f is **submodular**: $\forall S \subseteq T$

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

- Basic fact:

- If f_1, \dots, f_k are **submodular**, and $c_1, \dots, c_k \geq 0$

- then $\sum_i c_i \circ f_i$

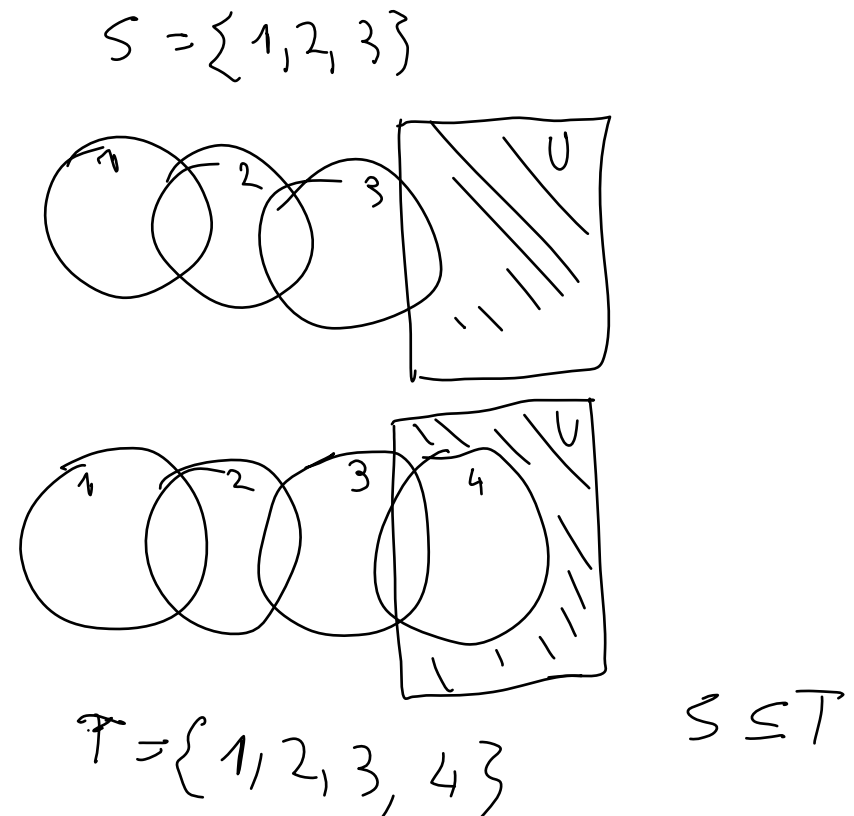
is also **submodular**

$f(S)$ is submodular

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

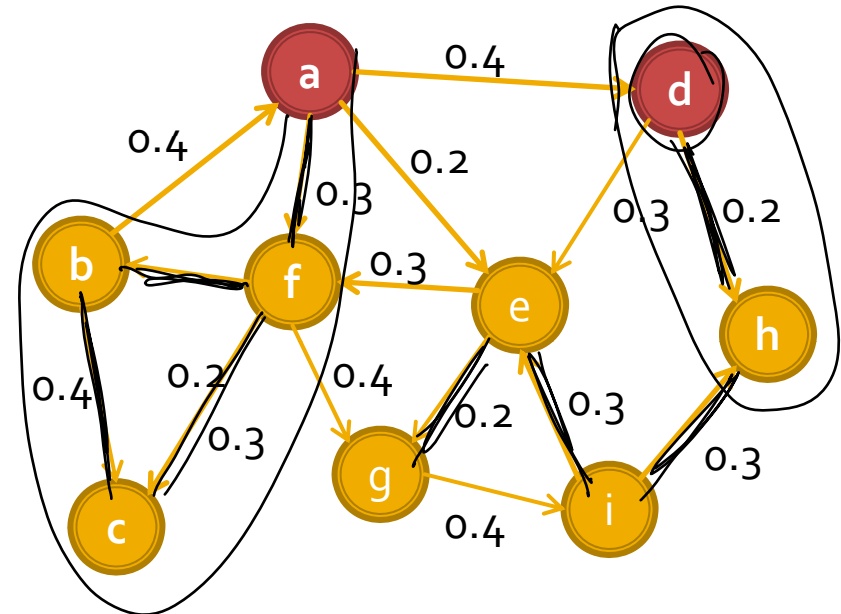
- A simple **submodular** function:

- Sets S_1, \dots, S_k
- $f(S) = |\cup_i S_i|$



Analysis: Alternative View

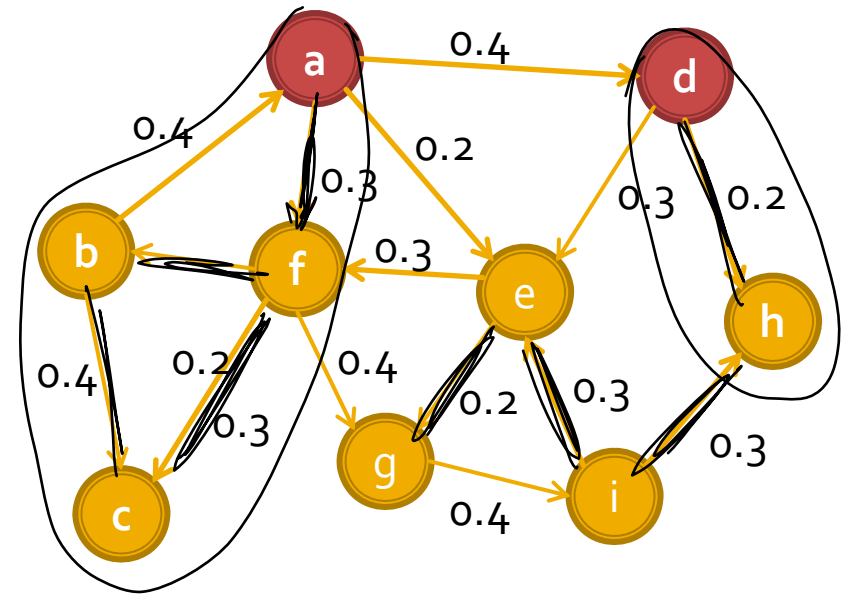
- Principle of deferred decision:
 - Generate randomness ahead of time



- Flip a coin for each edge to decide whether it will succeed when (if ever) it attempts to transmit
- Edges on which activation will succeed are **live**
- $f(S_i)$ = size of the set reachable by live-edge paths

Analysis: Alternative View

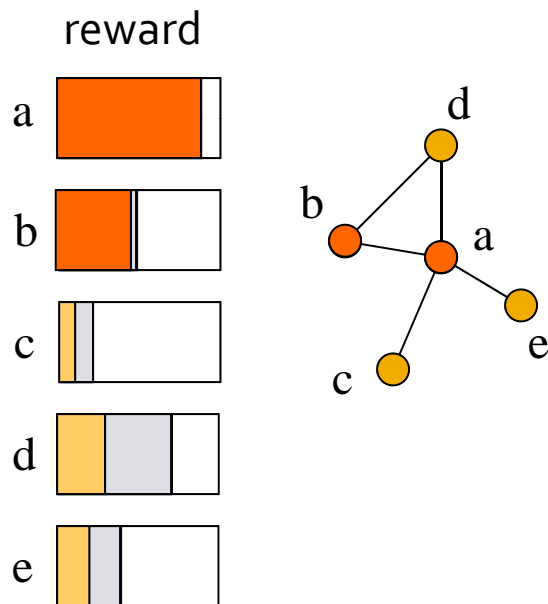
- Fix outcome i of coin flips
- Let $f_i(S)$ be size of cascade from S given these coin flips



- Let $f_i(v)$ = set of nodes reachable from v on **live-edge** paths
- $f_i(S)$ = size of union $f_i(v) \rightarrow f_i$ is **submodular**
- $f = \sum f_i \rightarrow f$ is **submodular**

Background: Submodular functions

Hill-climbing



Add sensor with highest
marginal gain

What do we know about optimizing
submodular functions?

- A hill-climbing (*i.e.*, greedy) is near optimal ($1-1/e$ ($\sim 63\%$) of optimal)
- But
 - Hill-climbing algorithm is **slow**
 - At each iteration we need to re-evaluate marginal gains
 - It scales as $O(nk)$

Speeding up hill climbing

- Observation:
Submodularity guarantees that marginal benefits **decrease** with the solution size

$$f(S_0 \cup \{x\}) - f(S_0) \geq f(S_1 \cup X) - f(S_1) \geq f(S_2 \cup X) - f(S_2)$$

Marginal gain $f(S_i \cup \{x\})$

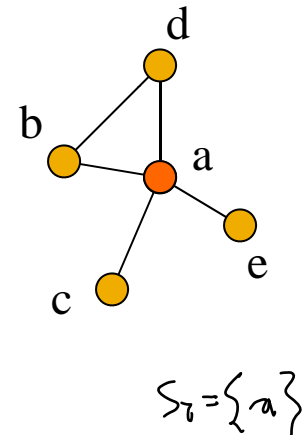


- Idea: exploit submodularity, doing **lazy evaluations!**

Lazy evaluation

- **Lazy hill-climbing:**
 - Keep an ordered list of marginal benefits b_i from previous iteration
 - Re-evaluate b_i **only** for top node
 - Re-sort and prune

Marginal gain

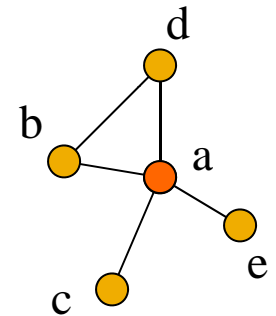
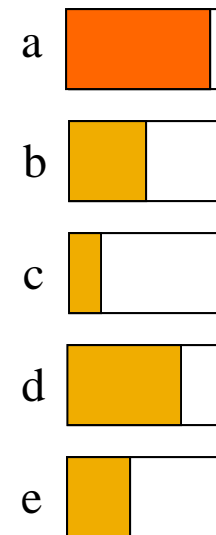


$$f(S_0 \cup \{a\}) - f(S_0)$$

Lazy evaluation

- **Lazy hill-climbing:**
 - Keep an ordered list of marginal benefits b_i from previous iteration
 - Re-evaluate b_i **only** for top node
 - Re-sort and prune

Marginal gain

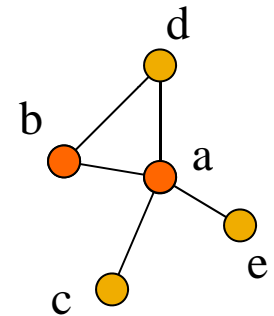
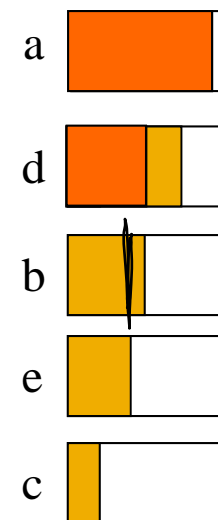


$$f(S_1 \cup N) - f(S_0) \stackrel{?}{=} f(S_0 \cup N) - f(S_0)$$

Lazy evaluation

- **Lazy hill-climbing:**
 - Keep an ordered list of marginal benefits b_i from previous iteration
 - Re-evaluate b_i **only** for top node
 - Re-sort and prune

Marginal gain



Problem: Water Network

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

