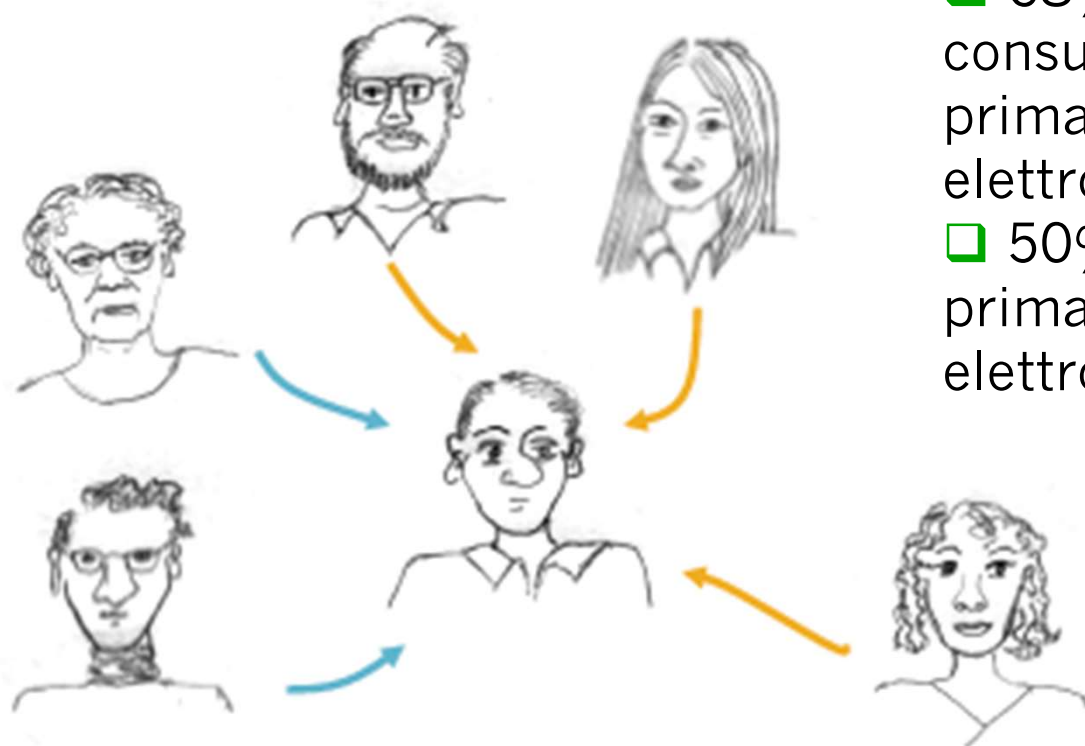


Massimizzare
l'influenza

Viral marketing

- Noi siamo influenzati più dai nostri amici che da persone che non conosciamo



- ❑ 68% dei consumatori consultano amici e familiari prima di comprare oggetti elettronici per la casa
- ❑ 50% fa delle ricerche online prima di comprare oggetti elettronici

Viral marketing

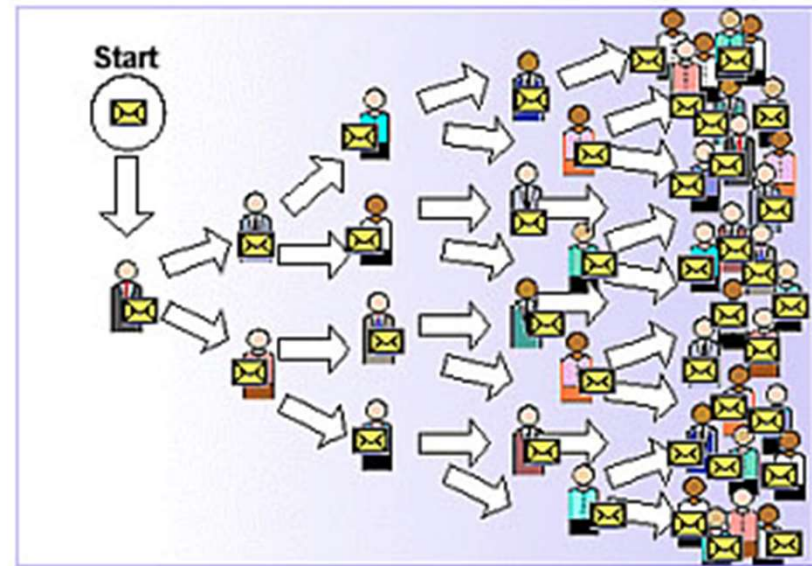
Identificare clienti
che possano
influenzare



Convincerli ad adottare un
prodotto – offrire essi
sconti o campioni gratuiti



Questi clienti consigliano
il prodotto ai loro amici



Come creare una “grande” cascade?

- Information epidemics:
 - quali sono gli utenti che influenzano di più?
 - quali sono i siti che consentono una grande diffusione?
 - Dove dovremmo fare pubblicità?

Massimizzare lo spread (diffusione)

- Supponiamo che invece di un virus, abbiamo un **oggetto** (prodotto, idea, video, etc.) che si propaga attraverso l'interazione (**contatto**)
 - **word of mouth propagation** (passaparola)
- una azienda che reclamizza un prodotto è interessata a **massimizzare lo spread** del prodotto nella rete
 - il santo Graal del “**viral marketing**”

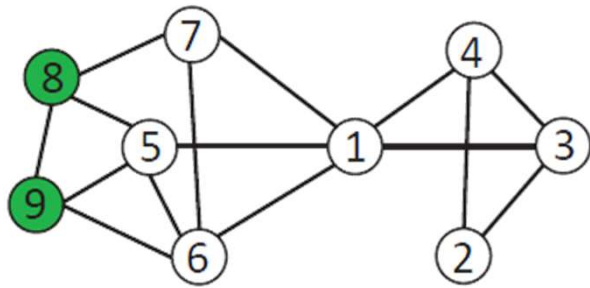
Problema: quali nodi dovremmo “**infettare**” così da massimizzare lo spread?

Contagio probabilistico

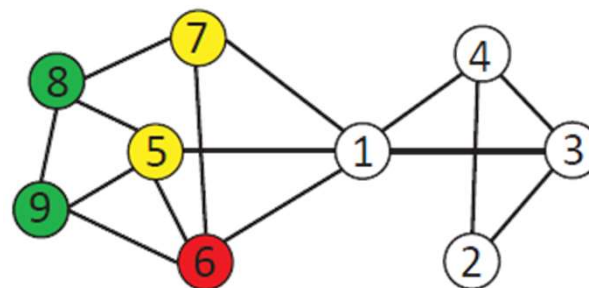
Independent cascade model

- Grafo direzionato $G=(V,E)$
- Ciascun nodo può essere **attivo** (ha il prodotto) o **inattivo** (non ha il prodotto)
- Il tempo procede in step.
- Al tempo **t**, ogni nodo **v** che è divenuto attivo al tempo **t-1** attiva un adiacente **w** non-attivo con probabilità p_{vw} . Se fallisce non può più riprovarci
 - un nodo ha solo una chance di rendere un adiacente attivo
 - generalizzazione del **modello SIR** (in quel caso tutte le probabilità erano uguali) con $t_I = 1$

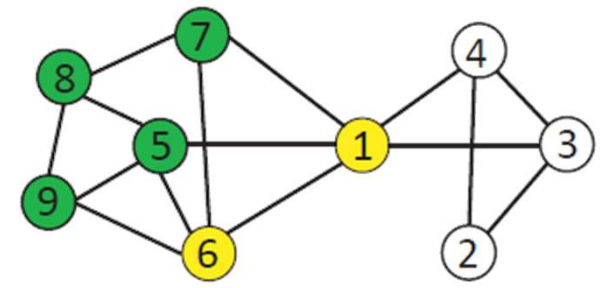
Independent cascade



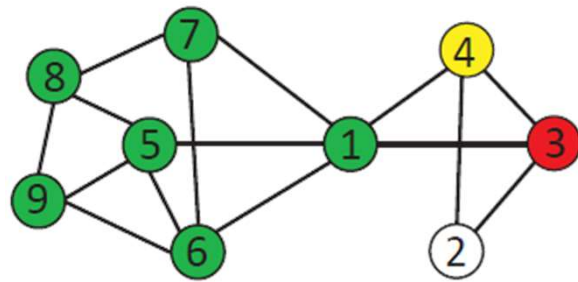
Step 0



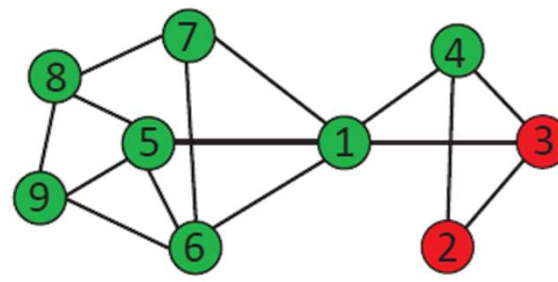
Step 1



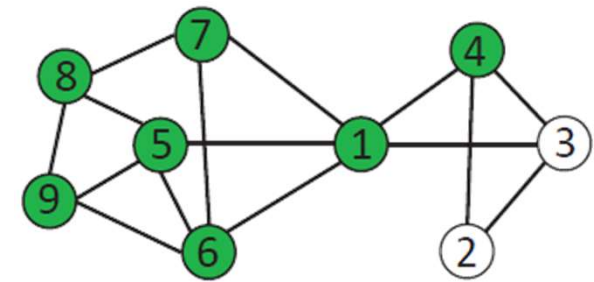
Step 2



Step 3



Step 4



Final Stage

Massimizzare l'influenza

Funzione di influenza: per un insieme S (target set), l'influenza $f(S)$ (spread) è il numero atteso di nodi attivi alla fine del processo di diffusione, se il prodotto è inizialmente posto nei soli nodi in S

Problema della massimizzazione dell'influenza

[Kempe, Kleinberg, Tardos 2003]:

Data una rete rappresentata da un grafo G ed un valore k , identificare un insieme S di k nodi nella rete che massimizzi $f(S)$.

$$\max_{S: |S|=k} f(S)$$

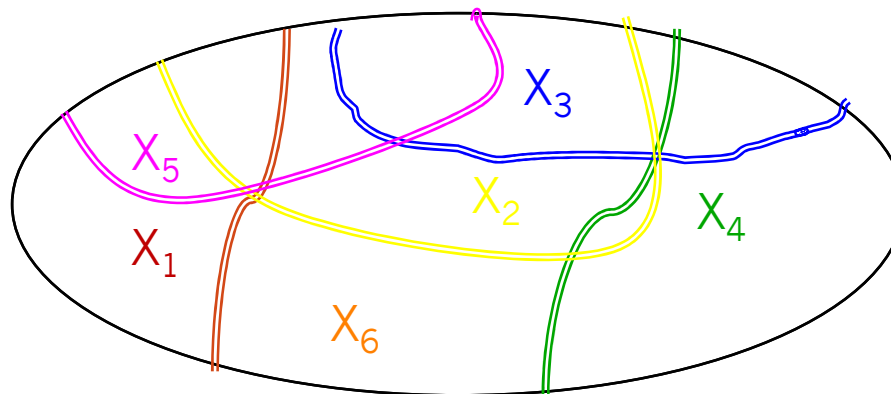
Massimizzare l'influenza è NP-hard

- Mostriamo che determinare il più influente target set è almeno difficile quanto il problema di determinare il set cover

Set cover problem (noto problema NP-complete)

Dato un universo $U = \{u_1, \dots, u_n\}$ ed insiemi $X_1, \dots, X_m \subseteq U$

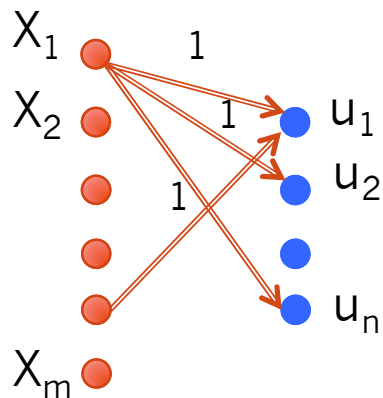
Ci sono k insiemi tra X_1, \dots, X_m tali che la loro unione è U ?



Massimizzare l'influenza è NP-hard (dimostrazione)

- Data un'istanza di set cover k , $U=\{u_1, \dots, u_n\}$, X_1, \dots, X_m
- Costruiamo un grafo direzionato bipartito $G=(V,E)$
 - $V= \{X_1, \dots, X_m\} \cup \{u_1, \dots, u_n\}$
 - $E=\{ (X_h, u_j) \mid u_j \in X_h \}$
 - $p_{X_h u_j}=1$ se $(X_h, u_j) \in E$

Nota: avendo scelto le probabilità tutte uguali a 1, stiamo in pratica considerando un'attivazione deterministica



Possibile provare che:
Esiste un target set S di size k con $f(S)=k+n$ iff esiste un set cover di size k

cattive e buone notizie

- Notizie cattive:
 - La massimizzazione dell'influenza è NP-complete
- Notizie buone:
 - esiste un algoritmo di approssimazione!
 - * l'algoritmo non trova una soluzione globalmente ottima per tutti gli input
 - * ma, proveremo che l'algoritmo non si comporta troppo male. Più precisamente, esso trova un target set S tale che $f(S) > 0,63 f(S_{OPT})$

L'algoritmo Greedy

- Il più semplice ed intuitivo algoritmo per selezionare S

Algoritmo GREEDY(G, k)

Parti con $S_0 = \emptyset$

Procedi in k step, for ciascuno step $i = 1 \dots k$

- scegli il nodo u che **massimizza** la funzione $f(S_{i-1} \cup \{u\})$
(il nodo che attiva il maggior numero di nodi in quello step)
- aggiungi u all'insieme S_{i-1} cioè poni $S_i = S_{i-1} \cup \{u\}$

- Calcolo di $f(S)$: effettua **simulazioni** Monte-Carlo ripetute del processo e prendi la media
- Quanto è buona questa soluzione comparata alla soluzione ottima?

Richiami: Algoritmi di approssimazione

Supponiamo di avere un problema di ottimizzazione, e sia

- X un'istanza del problema, sia
- $OPT(X)$ il valore della soluzione ottima per X , sia
- $ALG(X)$ il valore della soluzione di un algoritmo ALG per X

Nel nostro caso:

- * $X=(G,k)$ è l'istanza input,
- * $OPT(X)$ è lo spread $f(S_{OPT})$ della soluzione ottima S_{OPT} ,
- * $GREEDY(X)$ è lo spread $f(S)$ della soluzione dell'algoritmo Greedy

ALG è un algoritmo di approssimazione se il rapporto tra $OPT(X)$ e $ALG(X)$ è limitato

Richiami: Algoritmi di approssimazione

Per un problema di massimizzazione, l'algoritmo **ALG** è un α -approximation algorithm, per $\alpha < 1$, se per tutte le istanze X ,

$$\text{ALG}(X) \geq \alpha \text{OPT}(X)$$

- Questo significa che **ALG(X)** ha un valore di almeno $\alpha\%$ del valore ottimo
- α è l'**approximation ratio** dell'algoritmo
 - idealmente, vorremmo α sia una costante vicina a 1

Approximation ratio dell'algoritmo greedy per la massimizzazione dell'influenza

Teorema

L'algoritmo **GREEDY** ha un'approximation ratio $\alpha = 1 - 1/e$

$$\text{GREEDY}(G, K) \geq (1 - 1/e) \text{OPT}(G, k)$$

per tutte le istanze (G, k)

[Kempe, Kleinberg, Tardos 2003]

dimostrazione

La dimostrazione procede in due passi:

(1) prova che la funzione di influenza $f(\cdot)$ ha due proprietà

- f è **monotona** e **submodulare**

(2) determina il fattore di approssimazione dell'algoritmo GREEDY

Dimostrazione: passo (1)

La funzione di influenza $f(\cdot)$ ha due proprietà

- f è **monotona**

$$f(S) \leq f(T) \quad \text{se } S \subseteq T$$

- un nodo che si attiva, non può più disattivarsi, quindi ciò che riesco ad attivare con S posso sicuramente attivare (se non di più) con T

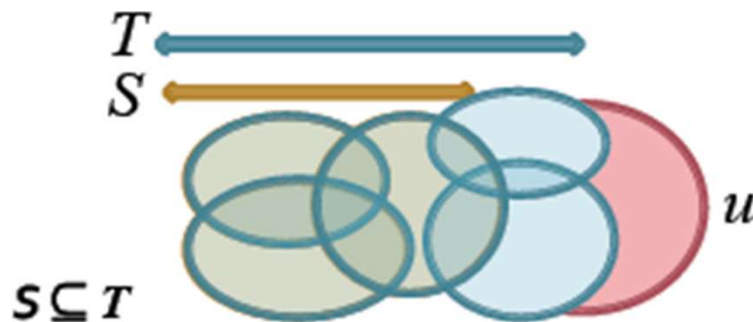
Dimostrazione: passo (1)

La funzione di influenza $f(\cdot)$ ha due proprietà

- f è **submodulare**

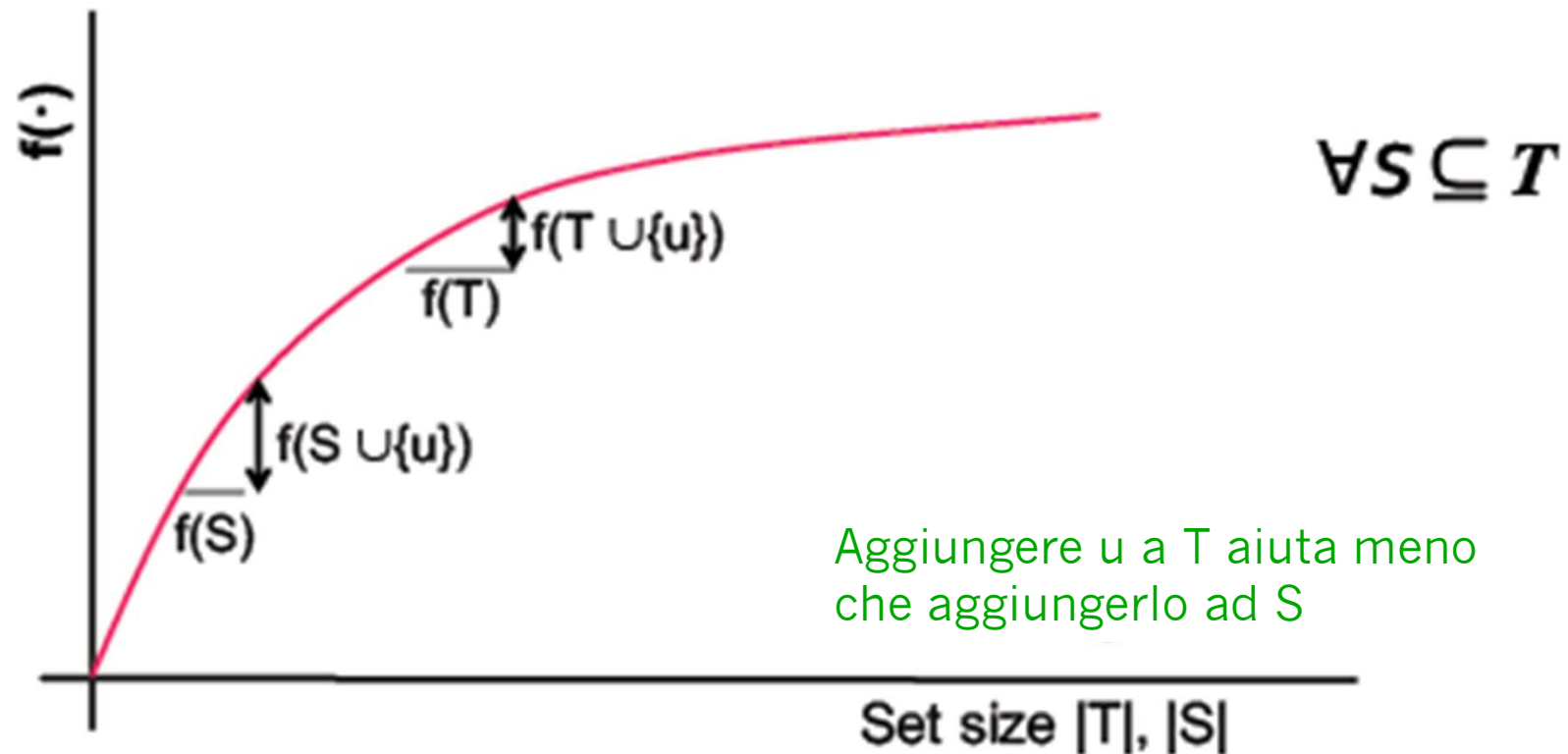
$$f(S \cup \{u\}) - f(S) \geq f(T \cup \{u\}) - f(T) \quad \text{se } S \subseteq T$$

- l'aggiunta di un nodo u al target set ha più effetto (più attivazioni) se il target set è **più piccolo** piuttosto che se fosse **più grande**.



più grande è l'insieme che ricopre più piccola è l'area di u che resta da coprire

submodularità



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

Background: funzioni submodulari

Fatto 1:

Se $f_1(x), \dots, f_k(x)$ sono submodulari,

e $c_i, \dots, c_k \geq 0$

allora $F(x) = \sum_i c_i f_i(x)$ è anch'essa submodulare

(combinazione lineare non negativa di funzioni submodulari e submodulare)

Fatto 2:

Dati gli insiemi X_u, \dots, X_v , dove

X_u nodi influenzati dall'iniziale nodo attivo u

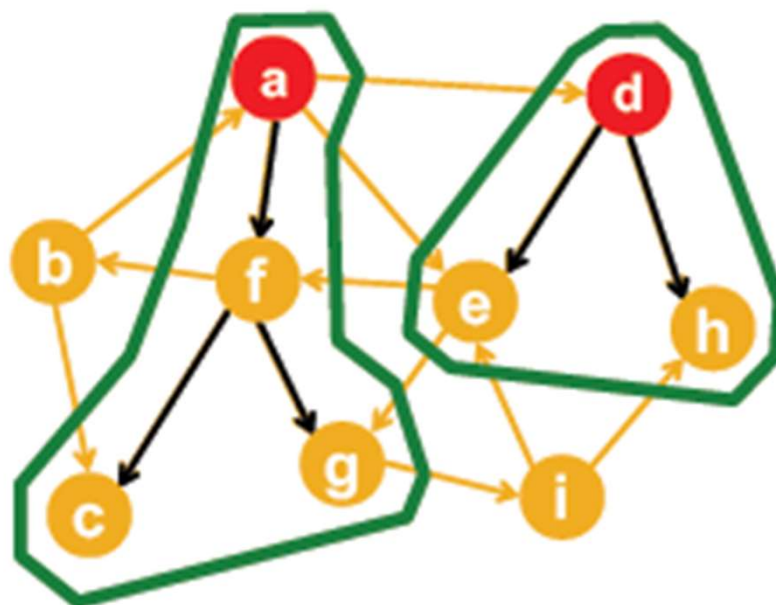
$f(S) = | \bigcup_{w \in S} X_w |$ è una funzione submodulare

$f(S)$ è submodulare

Strategia della dimostrazione:

Usiamo il fatto che la massimizzazione dell'influenza è un'istanza del problema del vertex cover:

- $f(S)$ è la size dell'unione dei nodi influenzati dai vertici attivi in S



Bisogna però stare attenti perché $f(S)$ è il risultato di un processo random.

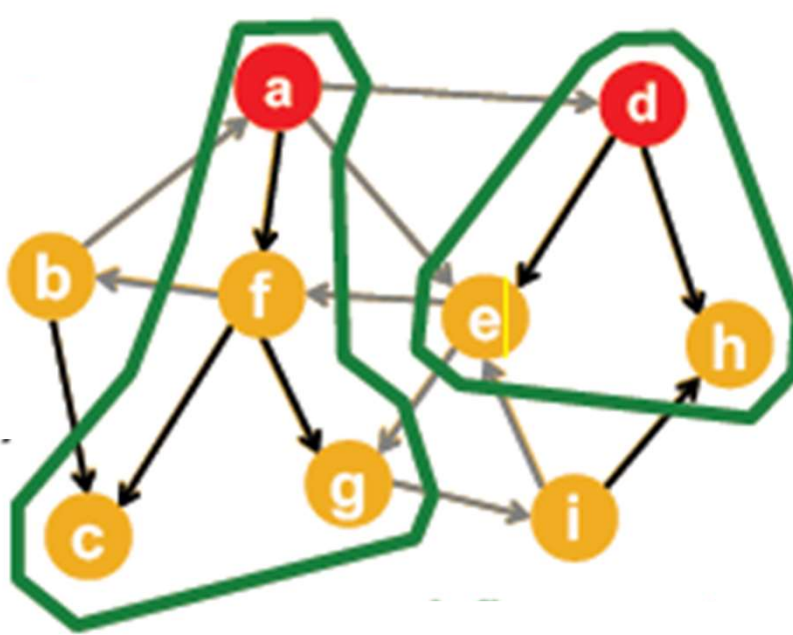
Useremo il **principio di decisione differita**

$f(S)$ è submodulare

Principio di decisione differita:

Lanciamo all'inizio la moneta per ciascun edge e memorizziamo per quali edge si ha il successo

- rimuoviamo gli edge per cui il lancio della moneta ha dato un insuccesso
- gli edge sopravvissuti andranno a costituire un grafo deterministico!



Bisogna ora individuare l'insieme X_u dei nodi influenzati da ciascun nodo u :
 X_u è l'insieme dei nodi raggiungibili da u lungo una path di edge sopravvissuti

$$X_a = \{a, f, c, g\}$$

$$X_b = \{b, c\}$$

$$X_c = \{c\}$$

$$X_d = \{d, e, h\}$$

.....

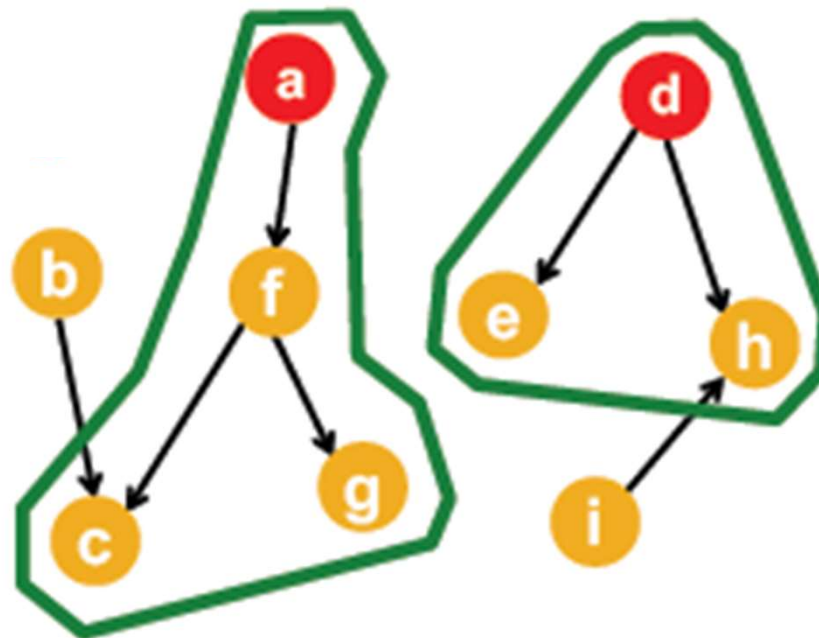
$f(S)$ è submodulare

X_u insieme dei nodi influenzati da ciascun nodo u :

- X_u è l'insieme dei nodi raggiungibili da u lungo una path di edge sopravvissuti

Quale è il valore di $f(S)$?

- $f(S)$ è la size dell'insieme di nodi raggiungibili da nodi in S lungo path di edge sopravvissuti



$$X_a = \{a, f, c, g\}$$

$$X_b = \{b, c\}$$

$$X_c = \{c\}$$

$$X_d = \{d, e, h\}$$

.....

$$f(\{a, b\}) = |\{a, f, c, g\} \cup \{b, c\}| = 5$$

$$f(\{a, d\}) = |\{a, f, c, g\} \cup \{d, e, h\}| = 7$$

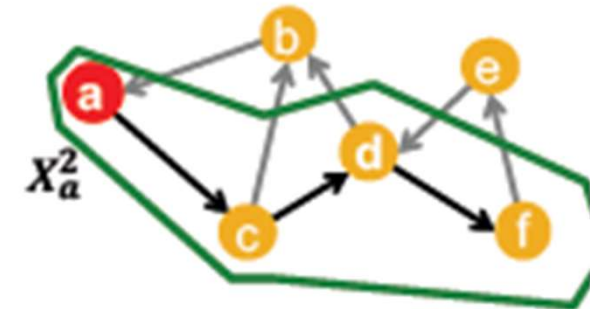
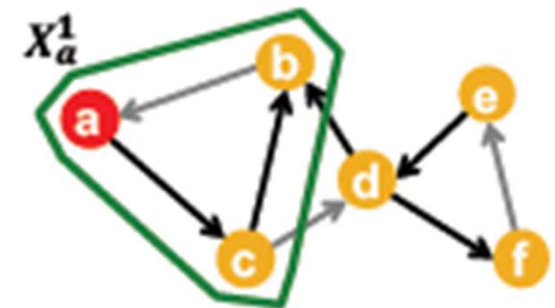
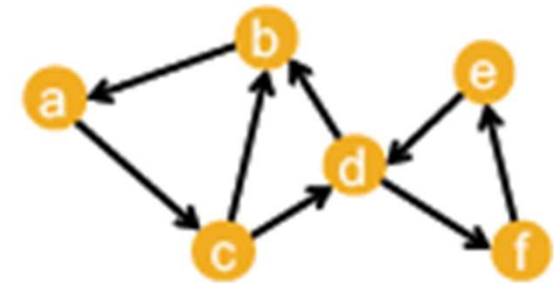
$f(S)$ è submodulare

Il valore di $f(S)$ dipende dal risultato del lancio di monete che ha determinato gli archi del grafo

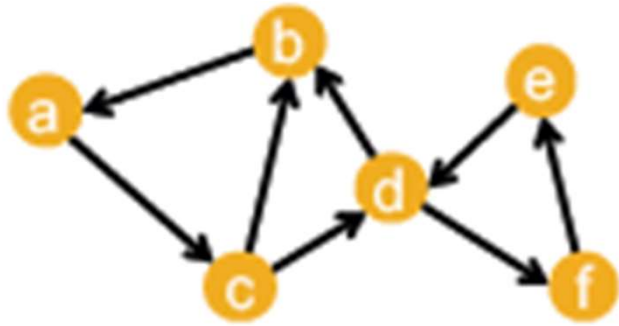
Ripetiamo più volte questo processo

Con l' i -simo lancio di monete, determiniamo

- X_a^i = insieme di nodi raggiungibili dal nodo a lungo path di edge sopravvissuti nell' i -simo lancio
- $f_i(S) = |\bigcup_{w \in S} X_w^i| \rightarrow f_i(S)$ è submodulare (Fatto 2)
- $f(S) = \sum_i f_i(S) \rightarrow f(S)$ è submodulare (combinazione lineare di funzioni submodulari - Fatto 1)



esempio

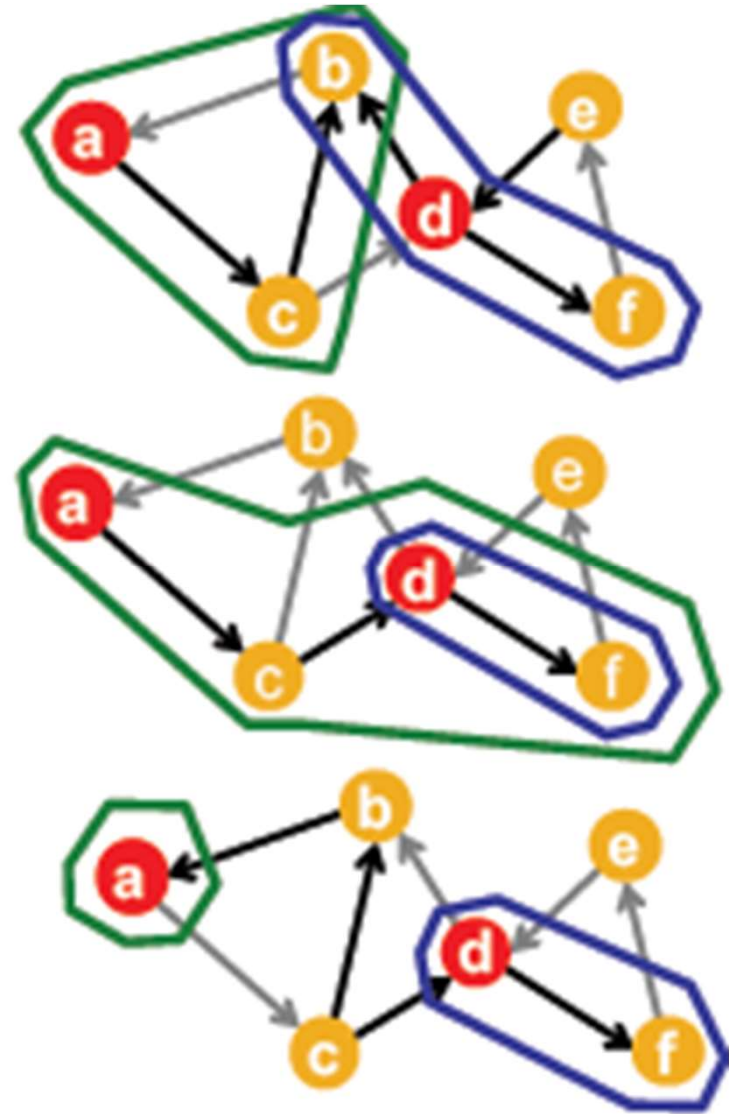


Consideriamo $S=\{a,d\}$

e tre lanci di monete e quindi
tre grafi, abbiamo

$$f_1(S) = 5 \quad f_2(S) = 4 \quad f_3(S) = 3$$

$$f(S) = \sum_i f_i(S) = 12$$



Dimostrazione : passo (2)

Abbiamo fatto vedere che la funzione $f(\cdot)$ usata nell'algoritmo GREEDY è **monotona** e **submodulare**, ci rimane da determinare il fattore di approssimazione dell'algoritmo GREEDY, cioè:

Il fattore di approssimazione dell'algoritmo GREEDY è $\alpha = 1 - 1/e$ cioè esso produce un insieme di attivazione S tale che

$$f(S) \geq (1 - 1/e) f(S_{OPT})$$

Questa soluzione greedy è **almeno 63%** della soluzione ottima

- È un bound nel caso peggiore
- Indipendentemente dai dati input, l'algoritmo greedy non fa mai peggio di $0.63 f(S_{OPT})$

Strategia della dimostrazione

Definiamo: guadagno marginale $\delta_i = f(S_i) - f(S_{i-1})$

La dimostrazione procede in 3 passi:

0) Lemma: $f(A \cup B) - f(A) \leq \sum_{j=1 \dots k} [f(A \cup \{b_j\}) - f(A)]$ dove $B = \{b_1, \dots, b_k\}$ e $f(\cdot)$ è submodulare

1) $\delta_{i+1} \geq 1/k [f(S_{OPT}) - f(S_i)]$

2) $f(S_{i+1}) \geq (1 - 1/k) f(S_i) + 1/k f(S_{OPT})$

3) $f(S_k) \geq (1 - 1/e) f(S_{OPT})$

Passo 0

$$f(A \cup B) - f(A) \leq \sum_{i=1 \dots k} [f(A \cup \{b_i\}) - f(A)]$$

dove $B = \{b_1, \dots, b_k\}$ e $f(\cdot)$ è submodulare

Dim.:

Passo 0

$$f(A \cup B) - f(A) \leq \sum_{i=1 \dots k} [f(A \cup \{b_i\}) - f(A)]$$

dove $B = \{b_1, \dots, b_k\}$ e $f(\cdot)$ è submodulare

Dim.:

- sia $B_i = \{b_1, \dots, b_i\}$, così abbiamo $B_1, B_2, \dots, B_k (=B)$ e $B_i = B_{i-1} \cup \{b_i\}$ e $B_0 = \emptyset$

Passo 0

$$f(A \cup B) - f(A) \leq \sum_{i=1 \dots k} [f(A \cup \{b_i\}) - f(A)]$$

dove $B = \{b_1, \dots, b_k\}$ e $f(\cdot)$ è submodulare

Dim.:

- sia $B_i = \{b_1, \dots, b_i\}$, così abbiamo $B_1, B_2, \dots, B_k (=B)$ e $B_i = B_{i-1} \cup \{b_i\}$ e $B_0 = \emptyset$
- $f(A \cup B) - f(A) = \sum_{i=1 \dots k} [f(A \cup B_i) - f(A \cup B_{i-1})]$



aggiungiamo e togliamo $f(A \cup B_1), f(A \cup B_2), \dots, f(A \cup B_{k-1})$

Passo 0

$$f(A \cup B) - f(A) \leq \sum_{i=1 \dots k} [f(A \cup \{b_i\}) - f(A)]$$

dove $B = \{b_1, \dots, b_k\}$ e $f(\cdot)$ è submodulare

Dim.:

- sia $B_i = \{b_1, \dots, b_i\}$, così abbiamo $B_1, B_2, \dots, B_k (=B)$ e $B_i = B_{i-1} \cup \{b_i\}$ e $B_0 = \emptyset$
- $$\begin{aligned} f(A \cup B) - f(A) &= \sum_{i=1 \dots k} [f(A \cup B_i) - f(A \cup B_{i-1})] \\ &= \sum_{i=1 \dots k} [f(A \cup B_{i-1} \cup \{b_i\}) - f(A \cup B_{i-1})] \end{aligned}$$

Passo 0

$$f(A \cup B) - f(A) \leq \sum_{i=1 \dots k} [f(A \cup \{b_i\}) - f(A)]$$

dove $B = \{b_1, \dots, b_k\}$ e $f(\cdot)$ è submodulare

Dim.:

- sia $B_i = \{b_1, \dots, b_i\}$, così abbiamo $B_1, B_2, \dots, B_k (=B)$ e $B_i = B_{i-1} \cup \{b_i\}$ e $B_0 = \emptyset$

- $$\begin{aligned} f(A \cup B) - f(A) &= \sum_{i=1 \dots k} [f(A \cup B_i) - f(A \cup B_{i-1})] \\ &= \sum_{i=1 \dots k} [f(A \cup B_{i-1} \cup \{b_i\}) - f(A \cup B_{i-1})] \\ &\leq \sum_{i=1 \dots k} [f(A \cup \{b_i\}) - f(A)] \end{aligned}$$

dalla submodularità $A \cup B_{i-1} \supseteq A$

Passo 1: valutiamo δ_{i+1}

$$f(S_{OPT}) \leq f(S_i \cup S_{OPT})$$

 dalla monotonia di f

Passo 1: valutiamo δ_{i+1}

$$f(S_{OPT}) \leq f(S_i \cup S_{OPT})$$


$$= f(S_i \cup S_{OPT}) - f(S_i) + f(S_i)$$

Passo 1: valutiamo δ_{i+1}

$$f(S_{OPT}) \leq f(S_i \cup S_{OPT})$$

$$= f(S_i \cup S_{OPT}) - f(S_i) + f(S_i)$$

$$\leq \sum_{j=1 \dots k} [f(S_i \cup \{t_j\}) - f(S_i)] + f(S_i)$$



dal Lemma precedente,
assumendo
 $S_{OPT} = \{t_1, \dots, t_k\}$

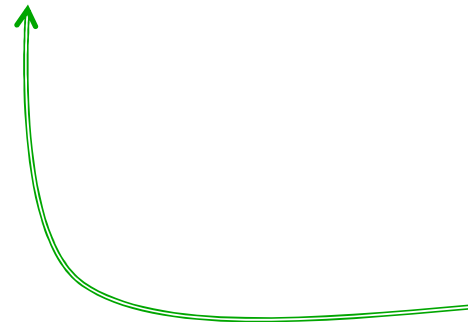
Passo 1: valutiamo δ_{i+1}

$$f(S_{OPT}) \leq f(S_i \cup S_{OPT})$$

$$= f(S_i \cup S_{OPT}) - f(S_i) + f(S_i)$$

$$\leq \sum_{j=1 \dots k} [f(S_i \cup \{t_j\}) - f(S_i)] + f(S_i)$$

$$\leq \sum_{j=1 \dots k} [\delta_{i+1}] + f(S_i)$$



$$S_{OPT} = \{t_1, \dots, t_k\}$$

considerando che la nostra scelta greedy porta a scegliere nello step j il vertice u per cui il valore di $f(\cdot)$ è migliore rispetto a qualunque altro vertice, si ha $f(S_i \cup \{t_j\}) \leq f(S_i \cup \{u\})$

$$\delta_{i+1} = f(S_{i+1}) - f(S_i)$$

Passo 1: valutiamo δ_{i+1}


$$\begin{aligned} f(S_{OPT}) &\leq f(S_i \cup S_{OPT}) \\ &= f(S_i \cup S_{OPT}) - f(S_i) + f(S_i) \\ &\leq \sum_{j=1 \dots k} [f(S_i \cup \{t_j\}) - f(S_i)] + f(S_i) \\ &\leq \sum_{j=1 \dots k} [\delta_{i+1}] + f(S_i) \\ &= k \delta_{i+1} + f(S_i) \end{aligned}$$

$$\text{Così} \quad \delta_{i+1} \geq 1/k [f(S_{OPT}) - f(S_i)]$$

Passo 2: quanto vale $f(S_{i+1})$?

$$\begin{aligned} f(S_{i+1}) &= f(S_i) + \delta_{i+1} \\ &\geq f(S_i) + 1/k [f(S_{OPT}) - f(S_i)] \end{aligned}$$

ma $\delta_{i+1} \geq 1/k [f(S_{OPT}) - f(S_i)]$



Passo 2: quanto vale $f(S_{i+1})$?

$$f(S_{i+1}) = f(S_i) + \delta_{i+1}$$

$$\geq f(S_i) + 1/k [f(S_{OPT}) - f(S_i)]$$

$$= (1 - 1/k) f(S_i) + 1/k f(S_{OPT})$$

Così $f(S_{i+1}) \geq (1 - 1/k) f(S_i) + 1/k f(S_{OPT})$

Passo 3: quanto vale $f(S_k)$?

Prima proviamo che

$$f(S_i) \geq [1 - (1 - 1/k)^i] f(S_{OPT})$$

Per induzione:

- $i = 0$: $f(S_0) = f(\emptyset) = 0$ e $[1 - (1 - 1/k)^0] f(S_{OPT}) = 0$

Passo 3: quanto vale $f(S_k)$?

Prima proviamo che

$$f(S_i) \geq [1 - (1 - 1/k)^i] f(S_{OPT})$$

Per induzione:

- $i = 0$: $f(S_0) = f(\emptyset) = 0$ e $[1 - (1 - 1/k)^0] f(S_{OPT}) = 0$
- Supponiamo che $f(S_i) \geq [1 - (1 - 1/k)^i] f(S_{OPT})$
- Consideriamo $i + 1$:

$$\begin{aligned} f(S_{i+1}) &\geq (1 - 1/k) f(S_i) + 1/k f(S_{OPT}) \\ &\geq (1 - 1/k) [1 - (1 - 1/k)^i] f(S_{OPT}) + 1/k f(S_{OPT}) \end{aligned}$$

Passo 3: quanto vale $f(S_k)$?

Prima proviamo che

$$f(S_i) \geq [1 - (1 - 1/k)^i] f(S_{OPT})$$

Per induzione:

- $i = 0$: $f(S_0) = f(\emptyset) = 0$ e $[1 - (1 - 1/k)^0] f(S_{OPT}) = 0$
- Supponiamo che $f(S_i) \geq [1 - (1 - 1/k)^i] f(S_{OPT})$
- Consideriamo $i + 1$:

$$\begin{aligned} f(S_{i+1}) &\geq (1 - 1/k) f(S_i) + 1/k f(S_{OPT}) \\ &\geq (1 - 1/k) [1 - (1 - 1/k)^i] f(S_{OPT}) + 1/k f(S_{OPT}) \\ &= [1 - (1 - 1/k)^{i+1}] f(S_{OPT}) \end{aligned}$$

Passo 3: quanto vale $f(S_k)$?

Così

$$f(S_k) \geq [1 - (1 - 1/k)^k] f(S_{OPT})$$

$$\geq (1 - 1/e) f(S_{OPT})$$

Ma $(1 - 1/k)^k \leq 1/e$



Considerato che $S = S_k$ e che quindi $f(S) = f(S_k)$ abbiamo

$$f(S) \geq (1 - 1/e) f(S_{OPT})$$

Come valutare $f(S)$?

- E' ancora un **problema aperto** valutare $f(S)$ in maniera efficiente
- Ma, si possono ottenere delle buone stime attraverso la simulazione
 - ripetendo il processo di diffusione abbastanza spesso

Qualità della soluzione

L'algoritmo GREEDY produce una soluzione S tale che

$$f(S) \geq (1 - 1/e) f(S_{OPT})$$

cioè $f(S) \geq 0,63 f(S_{OPT})$

Questo risultato è un bound indipendente dai dati

- questo è un bound nel caso peggiore
- indipendentemente dai dati input l'algoritmo GREEDY non farà mai peggio di $0,63 f(S_{OPT})$

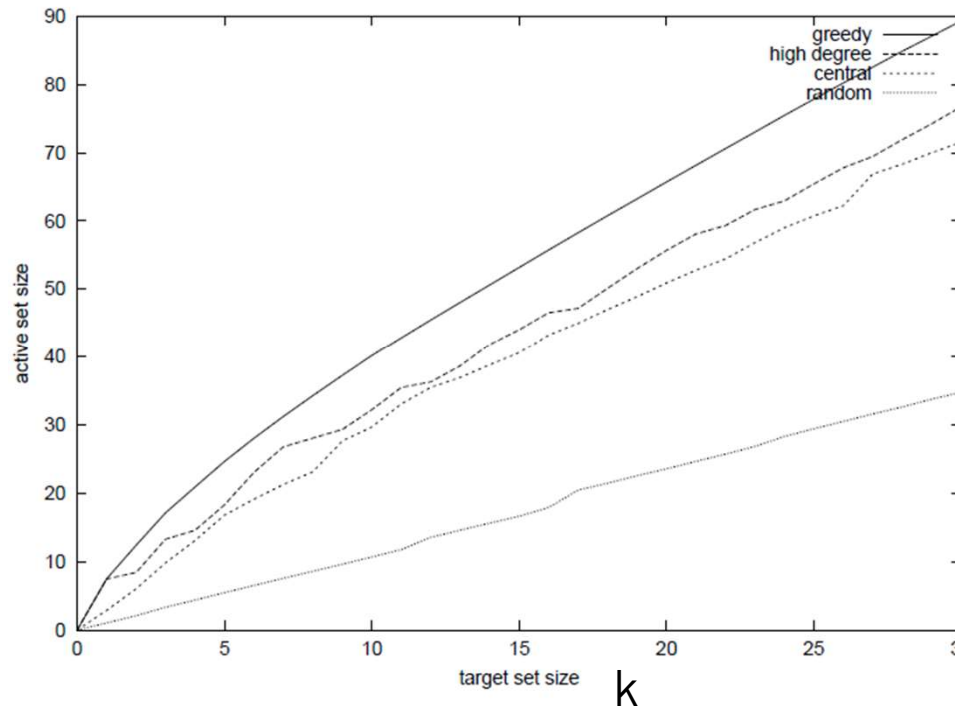
Esperimenti

- **Rete di collaborazione scientifica:** co-authorships in articoli presenti su arXiv riguardanti high-energy physics theory
 - 10748 nodi, 53000 edge
 - cascade process: spread di una nuova ricerca nell'area o una nuova terminologia
- **Independent Cascade Model:**
 - **Caso 1:** probabilità uniforme p per ciascun arco
 - **Caso2:** l'arco da v a w ha probabilità $1/\deg(w)$ di attivare w

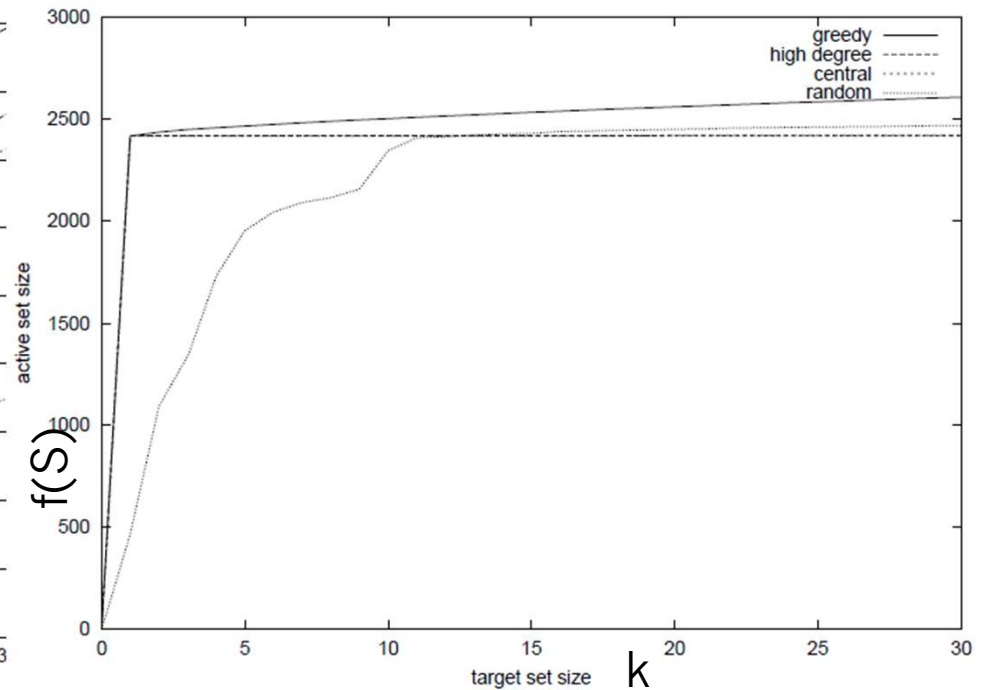
Esperimenti

- Simulare il processo 10000 volte
 - riscegliendo ogni volta gli archi in maniera random
- Confronto con 3 altre euristiche:
 - **degree centrality**: scegli il nodo con il grado maggiore
 - **distance centrality**: scegli il nodo al “centro” della rete
 - **Nodi a caso**: scegli un insieme di nodi a caso

Esperimenti



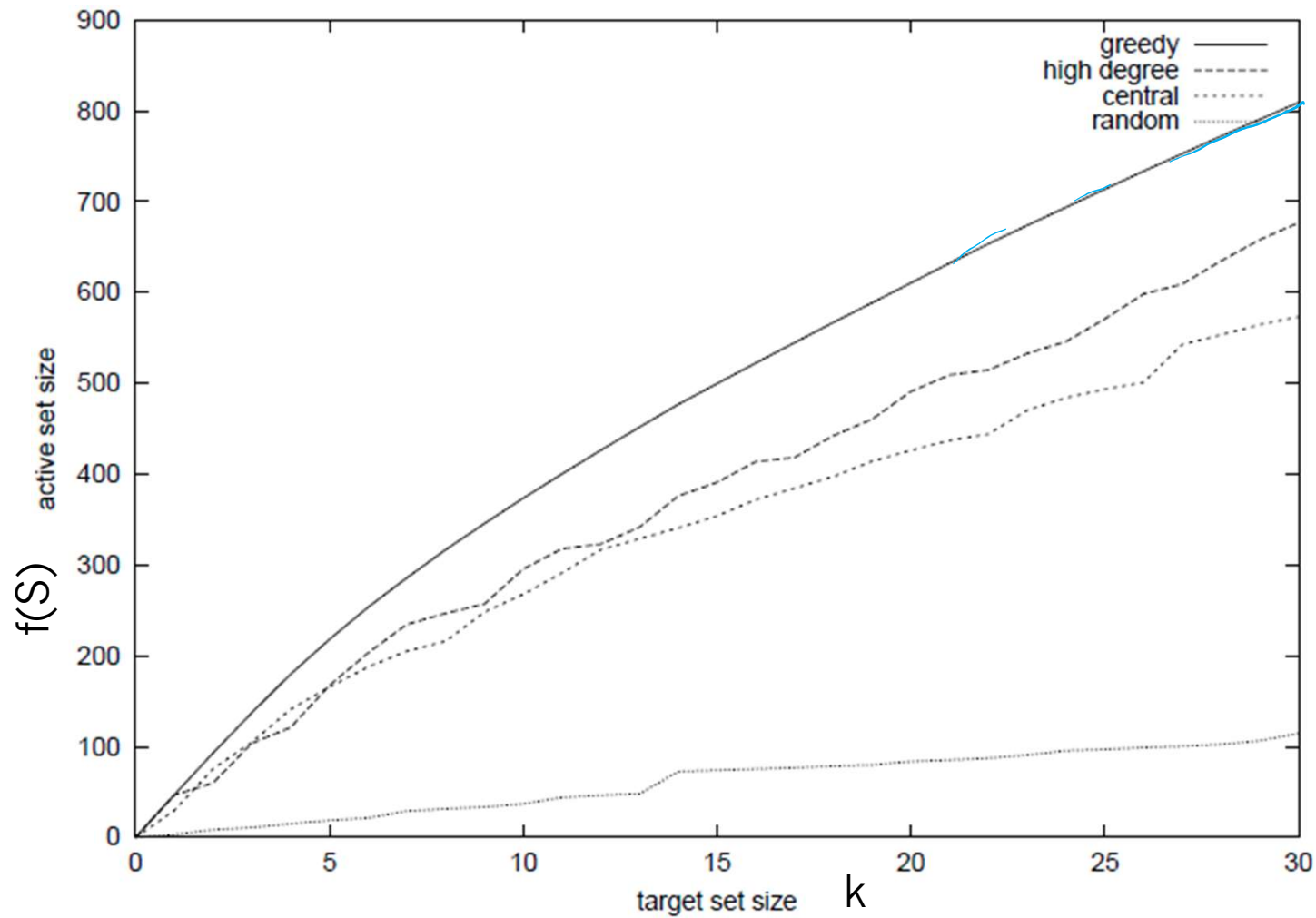
$p_{uv}=0.01$



$p_{uv}=0.10$

probabilità p_{uv} uniforme

Esperimenti



probabilità $p_{uv} = 1/\deg(v)$ non uniforme

Contagio deterministico

Linear threshold model

- Grafo direzionato $G=(V,E)$
- Ciascun nodo può essere **attivo** (ha il prodotto) o **inattivo** (non ha il prodotto)
- Ad ogni **arco direzionato** (v,u) ha associato un peso b_{vu} tale che

$$\sum_{v \in N_{in}(u)} b_{vu} \leq 1$$

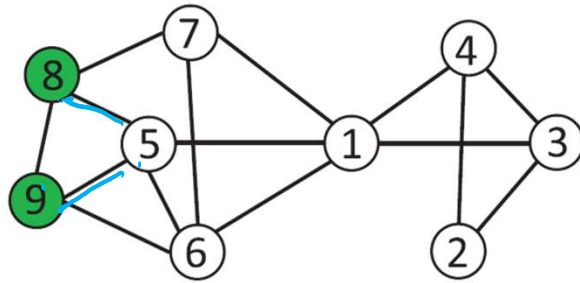
dove $N_{in}(u)$ = insieme degli adiacenti entranti di u

- Ogni nodo u ha un valore threshold t_u associato
- Il tempo procede in step.
- Al tempo t , un nodo **inattivo** u diventa **attivo** se

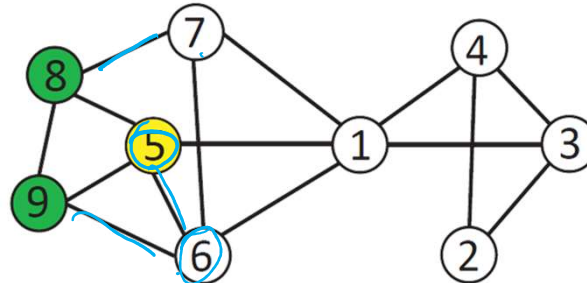
$$\sum_{v \text{ è attivo e } v \in N_{in}(u)} b_{vu} \geq t_u$$

Linear threshold model

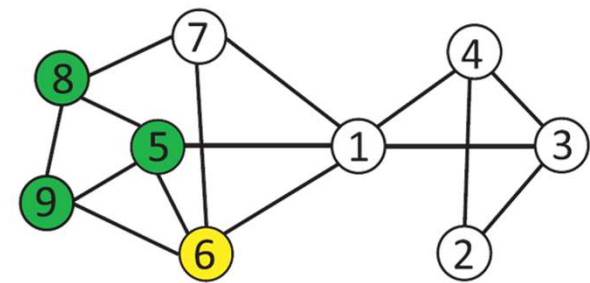
threshold a maggioranza



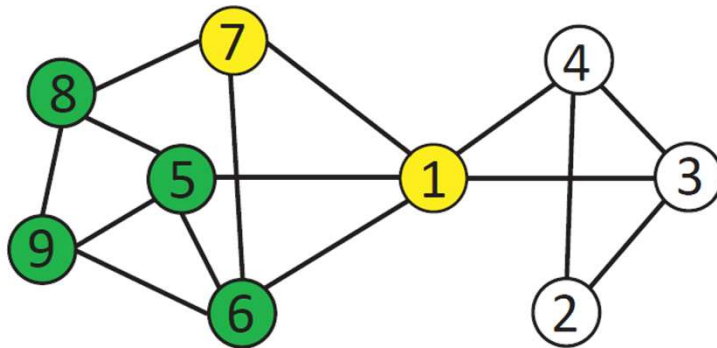
Step 0



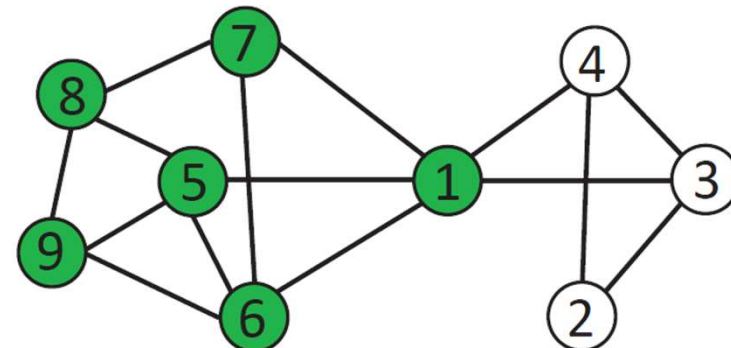
Step 1



Step 2



Step 3



Final Stage

Massimizzare l'influenza

- [Kempe, Kleinberg, Tardos 2003] hanno mostrato che anche nel linear threshold model,
 - la funzione $f(S)$ è submodulare
 - l'algoritmo Greedy ottiene $(1-1/e)$ approssimazione