# UNIVERSITÀ DEGLI STUDI DI SALERNO


## DIPARTIMENTO DI INFORMATICA

Corso di Laurea Magistrale in Informatica


## Relazione Reti Sociali

# Targeting with Partial Incentives


**Professoresse**

Luisa Gargano

Adele A. Rescigno

**Gruppo di progetto**

Francesco Manzo

Gerardo Manzo

Giuseppe Napoli



Anno Accademico 2019/2020

# 1 Introduzione

In questa relazione viene presentata una implementazione dell'algoritmo *Targeting with Partial Incentives*, descritto in "Whom to Befriend to Influence People" [3].

L'algoritmo è stato eseguito sul dataset *gemsec-Facebook* [2]. Il dataset contiene dati collezionati dalle pagine Facebook *verificate* caratterizzate dalla *spunta blu* accanto al nome; esso è suddiviso in categorie e per il nostro scopo abbiamo scelto di utilizzare la categoria 'Public Figures' contenente 11565 nodi e 67114 archi. Ogni nodo del grafo rappresenta una pagina mentre gli archi i like reciproci tra di esse. Queste pagine rappresentano individui, entità e organizzazioni con elevata visibilità. Per questo le aziende potrebbero essere interessate per le loro campagne pubblicitarie. Un'azienda potrebbe scegliere, ad esempio, un certo personaggio pubblico come sponsor affinché la sua campagna abbia successo. Questo personaggio come qualsiasi altra entità legata ad una pagina verificata ha di per sè un gran numero di like provenienti da persone e pagine comuni. Scegliere quindi una pagina a cui numerose altre pagine verificate hanno dato un like permetterebbe di raggiungere una platea di utenti ancora più grande. Oppure, un'azienda potrebbe scegliere una pagina relativa ad un sito web per pubblicizzare un prodotto e promettere al sito una percentuale del costo di esso per ogni utente che acquista il prodotto tramite tale sito.

# 2 Descrizione del problema

Data una rete sociale rappresentata da grafo $G = (V, E)$, dove $V$ è l'insieme dei nodi della rete, $E$ è l'insieme degli archi che rappresentano i collegamenti tra i nodi della rete, ed una threshod function $t : V \to N$ dove $t(v)$ indica il numero di adiacenti attivi del nodo $v$ necessari ad attivare $v$, si vuole trovare un insieme di nodi che sia in grado di influenzare l'intera rete.

Un **assegnamento di incentivi parziali** ai vertici del grafo $G$ con $V = v_1, ..., v_n$ è un vettore $s = (s(v_1), ..., s(v_n))$, dove $s(v) \in \{0, 1, 2, ...\}$ rappresenta la quantità di influenza applicata al nodo $v \in V$.

Un **processo di attivazione** in $G$ che inizia con un vettore di incentivi è una sequenza $Active[s, 0] \subseteq Active[s, 1] \subseteq ... \subseteq Active[s, l]... \subseteq V$ di sottoinsiemi di

vertici con:

- $Active[s, 0] = \{v : s(v) > t(v)\}$,
- $Active[s, l] = Active[s, l-1] \cup \{u : |N(u) \cap Active[s, l-1]| \geq t(u) - s(u)\}$, $\forall l > 0$

Un **vettore target** $s$ è un'assegnazione di incentivi parziali che innesca un processo di attivazione che influenza l'intera rete, cioè tale che $Active[s, l] = V$ per un $l \geq 0$. L'obiettivo è trovare il vettore $s$ che minimizza il totale degli incentivi per avere $Active[0, \infty] = V$, cioè un vettore $s$ che minimizzi:

$$C(s) = \sum_{v \in V} s(v)$$

# 3 Implementazione

L'algoritmo presentato in [3] è stato implementato utilizzando Python 3.6 e la libreria SNAP [1] per la manipolazione dei grafi.

Si utilizza un modello di attivazione con threshold. Dato un grafo non direzionato $G = (V, E)$, una threshold function $t : V \to N$ e una distribuzione di probabilità associata agli archi di $G$, $p : E \to [0, 1]$:

1. Applicare il *principio di decisione differita*: per ogni arco $e$ del grafo viene generato un numero pseudocasuale $x$ compreso tra 0 e 1. Se $x < p(e)$ (cioè il nodo infetta con una probabilità inferiore rispetto a quella richiesta), allora l'arco $e$ viene rimosso dal grafo.
2. Eseguire l'algoritmo sul grafo ottenuto: il grafo è dato in input all'algoritmo, che calcola la soluzione. Questa procedura è iterata 10 volte, e poi viene calcolata la dimensione media delle soluzioni ottenute.

Il progetto è quindi suddiviso in 3 file:

- `TPI.py`, contiene l'implementazione dell'algorimto TPI.

- `deferred_decision.py`, implementa il processo di decisione differita. Per le probabilità è possibile scegliere tra distribuzione uniforme e distribuzione normale.
- `main.py`, per l'esecuzione dell'algoritmo e la generazione dei grafici.

# 4 Risultati

I risultati ottenuti vengono rappresentati mediante i seguenti grafici. Sulle ascisse sono riportati i valori di trheshold utilizzati, mentre sulle ordinate è riportata la siza delle soluzioni trovate dall'algoritmo.

In Figura 1 è mostrata l'esecuzione dell'algoritmo, dopo aver applicato il principio di decisione differita al grafo, utilizzando thresholds costanti per tutti i nodi. In particolare, si è partiti assegnando ad ogni nodo una threshold pari a 1, e successivamente si è incrementata la threshold ad ogni esecuzione. Dato il gran numero di nodi e archi, l'incremento avviene con uno step pari a 2, e ci si è fermati al valore 11, che corrisponde al grado medio.

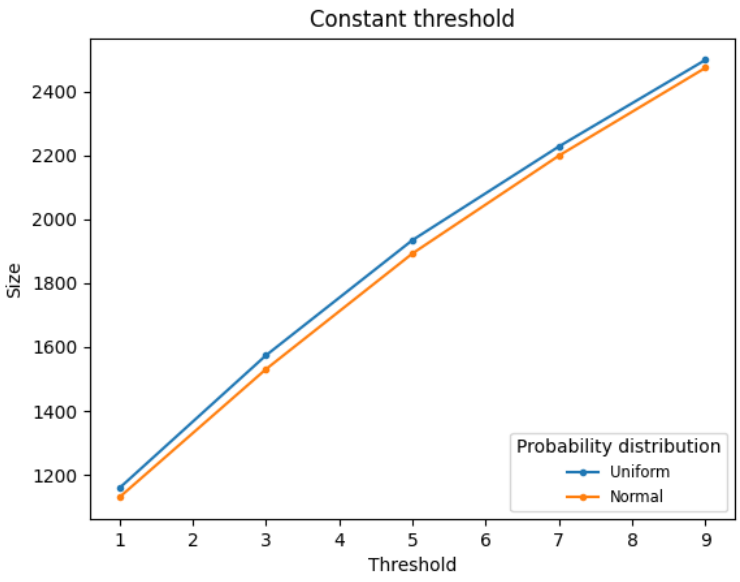


Figura 1: Constant thresholds

In Figura 2 è invece mostrata l'esecuzione dell'algoritmo utilizzando thresholds proporzionali al grado del nodo. La strategia di esecuzione è la stessa, ma le

threshold stavolta sono calcolate tenendo in considerazione il grado del nodo:

$$t(v) = \frac{deg(v) * i}{iter}$$

dove:

- $t(v)$ è la threshold assegnata al nodo $v$
- $deg(v)$ è il grado del nodo $v$
- $i$ è l'iterazione attuale e $iter$ è il numero di iterazioni totali (ovvero, il grado medio)

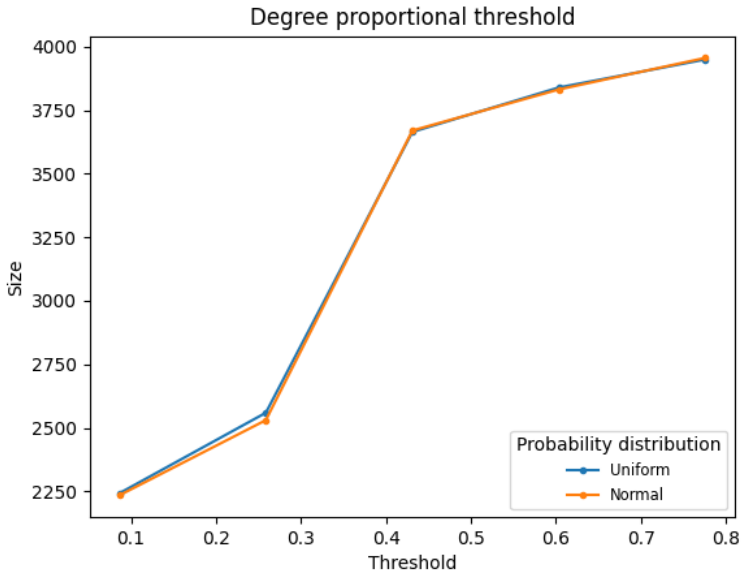


Figura 2: Degree proportional thresholds

In entrambi i casi vengono mostrate i risultati utilizzando sia la distribuzione uniforme che la distribuzione normale per la generazione delle probabilità assegnate agli archi. Per quanto riguarda *thresholds costanti* la size aumenta linearmente, con *thresholds proporzionali* abbiamo un salto, le size sono già alte a valori bassi per le threshold.

# Riferimenti bibliografici

[1] URL: `http://snap.stanford.edu/`.

[2] URL: http://snap.stanford.edu/data/gemsec-Facebook.html.

[3] Gennaro Cordasco, Luisa Gargano, Manuel Lafond, Lata Narayanand, Adele A. Rescigno, Ugo Vaccaro e Kangjang Wu. ≪Whom to Befriend to Influence People≫. In: *Theoretical Computer Science* 810 (2020), pp. 26–42.