Confronto dell'algoritmo TPI probabilistico rispetto al TPI deterministico sul costo totale degli incentivi

Vincenzo Russo, *Studente, Università degli Studi di Salerno.*

I. Introduzione

La diffusione dell'influenza nelle reti sociali è un fenomeno ampiamente studiato in contesti come il marketing virale e le scienze sociali. Un problema associato a questo fenomeno è il Target Set Selection [2], esso consiste nel trovare il minimo insieme dei nodi da influenzare in modo tale da causare un effetto cascade nella rete.
Una variante di questo problema è il Targeting with Partial Incentives [1] il cui obiettivo è trovare l'assegnamento di incentivi che minimizza il costo totale per influenzare l'intera rete. L'obiettivo di questo progetto è confrontare l'algoritmo TPI in versione deterministica rispetto allo stesso ma in versione probabilistica sul costo totale degli incentivi erogati.

A. Dataset

Il dataset utilizzato per confrontare i nostri algoritmi è la rete di collaborazione Arxiv GR-QC (General Relativity and Quantum Cosmology) [4] proveniente dall'archivio scientifico arXiv [3], esso tratta delle collaborazioni scientifiche tra autori che hanno sottomesso i loro paper nella categoria di Relatività Generale e Cosmologia Quantistica.
Il grafo, non orientato, conterrà un arco da i a j se l'autore i_{-esimo} è co-autore di un paper con l'autore j_{-esimo}. Se un articolo contiene k autori si ottiene un grafo fortemente connesso su k nodi. La struttura dati da noi esaminata è composta da 5242 nodi e 14496 archi. il suo grado medio è 5.53, il grado massimo è 81, il diametro è 13 e la rete contiene 48285 triangoli. Un esempio di analisi della rete nel contesto scientifico è cercare di capire quali sono gli autori più influenti in una rete di collaborazione scientifica.

II. Definizione del problema

Ciascun nodo $v \in V$ è associato con un valore non negativo $c(v)$, questo valore rappresenta il costo per convincere il membro v della rete ad adottare un dato prodotto o comportamento.
Data una rete sociale rappresentata dal grafo $G = (V, E)$, dove V, E rappresentano rispettivamente l'insieme dei nodi e degli archi del grafo, $t : V \to N$ è la threshold function ovvero il numero di nodi adiacenti attivi del nodo v necessari ad attivare v. $C : V \to N$ è la funzione che assegna i costi a ciascun nodo. si desidera trovare un target set $S \in V$ che minimizzi il costo $\mathrm{C(S)} = \sum_{v \in S} c(v)$ nell' intera rete G.

Un assegnamento di incentivi parziali sui nodi di una rete $G = (V, E)$ con $V = v_1, v_2, \ldots, v_n$ nodi, consiste nell'individuare un vettore $s = (s(v_1), s(v_2), \ldots, s(v_n))$ dove $s(v) \in \{0, 1, 2, \ldots\}$ rappresenta la quantità di influenza iniziale applicata su ciascun nodo $v \in V$.

Un processo di attivazione in G che inizia con un vettore di incentivi s è una sequenza Active $[\mathbf{s}, 0] \subseteq \mathrm{Active}[\mathbf{s}, 1] \subseteq \ldots \subseteq$ Active $[\mathbf{s}, \ell] \subseteq \ldots \subseteq \mathrm{V}$ di sottinsiemi di vertici con:

- Active $[\mathbf{s}, 0] = \{v : s(v) \geq t(v)\}, \quad \forall \ell > 0,$
- Active $[\mathbf{s}, \ell] =$ Active $[\mathbf{s}, \ell - 1] \cup \{\mathrm{u} :| \mathrm{N(u)} \cap$ Active $[\mathbf{s}, \ell - 1] \mid \quad \geq t(u) - s(u)\}.$

Un vettore target s è un assegnamento di incentivi parziali che innesca un processo di attivazione influenzando l'intera rete, tale che Active $[\mathbf{s}, \ell] = V$ per qualche $\ell > 0$.
L'obbiettivo è trovare il target vector s che minimizza gli incentivi totali per avere Active $[\mathbf{s}, \infty] = \mathrm{V}$.

III. Implementazione dell'algoritmo

L'algoritmo presentato in [1] è stato implementato utilizzando il linguaggio Python nella versione 3.7 [5] e la libreria SNAP [6] per la manipolazione delle strutture dati a grafo. Viene utilizzato 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 $p : E \to [0, 1]$:

- ***Applicare il principio di decisione differita:*** per ogni arco e del grafo G viene generato un numero pseudocasuale $x \in [0, 1]$. Se $x < p(e)$ (cioè il nodo infetta con una probabilità inferiore rispetto a quella richiesta), allora l'arco e viene rimosso dal grafo.
- ***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 contiene 3 files:

1) ***main.py:*** contenente l'esecuzione dell'algoritmo e la generazione dei grafici;
2) ***TPI.py:*** contenente l'implementazione dell'algoritmo TPI;
3) ***decisione_differita.py:*** contenente il processo di decisione differita. Per le probabilità con la possibilità di scegliere tra distribuzione uniforme e distribuzione normale.

IV. Risultati

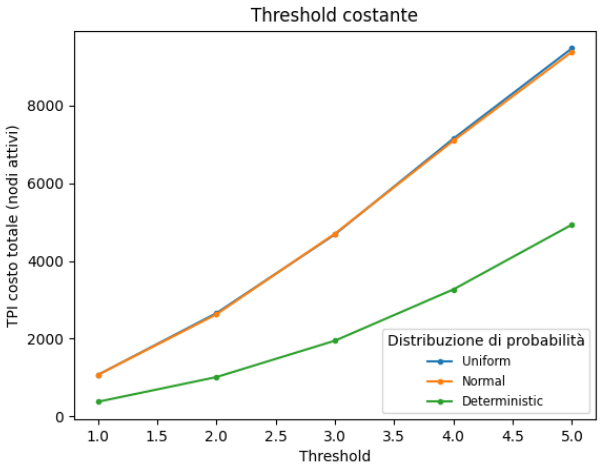

Fig. 1: Threshold costante

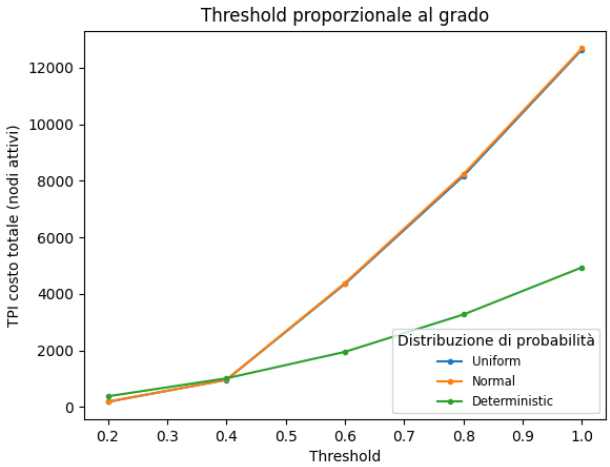

Fig. 2: Threshold proporzionale al grado

Sulle ascisse vengono riportati i valori di threshold utilizzati, mentre sulle ordinate viene riportata la somma dei costi degli incentivi applicata ai nodi attivi, forniti dall'algoritmo TPI. Si nota che in entrambe le figure le due funzioni di distribuzione ottengono risultati molto simili.

In [Fig. 1] viene confrontata l'esecuzione dell'algoritmo, con e senza il principio di decisione differita utilizzando thresholds t costanti per tutti i nodi, $t \in \{1, 2, 3, 4, 5\}$. Ci si è fermati al valore 5 , leggermente inferiore al grado medio 5.53.
É possibile notare che con la decisione differita sono necessari costi più alti per influenzare la rete, rispetto al caso deterministico senza decisione differita, ad esempio per il caso $t = 4$ è necessario un costo di circa 2000 per influenzare la rete, mentre con la decisione differita è necessario un costo di 6200.

In [Fig. 2] è invece mostrata l'esecuzione dell'algoritmo utilizzando thresholds proporzionali al grado del nodo. Per i valori di thresholds $\{0.2, 0.4\}$ è rappresentato un andamento comparabile dei costi sia per il deterministico che il non deterministico. Con una threshold 0.6 la somma dei costi dell'algoritmo non deterministico aumenta bruscamente, mentre il deterministico aumenta in maniera più regolare, questa differenza tra le due varianti dell'algoritmo tende ad accentuarsi all'aumentare della threshold proporzionale $\{0.8, 1\}$.

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 approssimato per difetto).

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.

V. Conclusioni

Come visto in precedenza è necessario un costo maggiore per influenzare una rete su cui viene applicato l'algoritmo non deterministico utilizzando la decisione differita. Anche se per thresholds di attivazione basse i costi tendono ad essere simili, all' aumentare di queste la differenza tra i costi dell'algoritmo in versione deterministica e non deterministica varia sensibilmente.

References

[1] "Whom to befriend to influence people", Gennaro Cordasco, Luisa Gargano, Manuel Lafond, Lata Narayanan, Adele A. Rescigno, Ugo Vaccaro, Kangkang Wu. Theoretical Computer Science 810: 26-42 (2020).

[2] "Discovering Small Target Sets in Social Networks: A Fast and Effective Algorithm", Gennaro Cordasco, Luisa Gargano, Marco Mecchia, Adele A. Rescigno, Ugo Vaccaro. Algorithmica 80(6): 1804-1833 (2018)

[3] https://arxiv.org/

[4] https://snap.stanford.edu/data/ca-GrQc.html

[5] https://www.python.org/downloads/release/python-370/

[6] http://snap.stanford.edu/