# Progetto di Social Computing

## Analisi delle dipendenze del pacchetto `@angular/cli` di `npm`

Jacopo Danielis [162553], Massimiliano Di Marco [144714],
Aurora Marzinotto [162556], Enrico Peressin [163503]

Corso di Social Computing,
a.a. 2024/2025

2025/01/07

# Indice

# 1. Introduzione

## 1.1. Sommario

L'obiettivo di questo progetto è sviluppare un software in grado di generare e analizzare il **grafo delle dipendenze** di un qualsiasi pacchetto `npm`, visualizzandolo graficamente e calcolandone alcuni parametri.
In un grafo delle dipendenze, ogni nodo rappresenta una libreria o un pacchetto software, mentre ogni arco (diretto) indica una relazione di dipendenza. Si ottiene così una **rete** che evidenzia i componenti che dipendono da altri per funzionare correttamente.
Nella seconda fase del progetto, il grafo verrà trasformato in indiretto, con l'aggiunta di nodi e archi artificiali, per calcolare metriche sui grafi risultanti. Verranno così confrontate le misure per dedurre delle proprietà afferenti ai grafi creati.

## 1.2. Descrizione generale del lavoro

Le funzionalità che il software deve rispettare sono riportate di seguito:

1. Costruzione del grafo delle dipendenze del *seed* tramite `requests` e `NetworkX`.
2. Trasformazione del grafo in indiretto con aumento di nodi e archi secondo la tecnica del *preferential attachment*.
3. Come bonus l'uso di una variante del *prefential attachment*: il *random walk*.
4. Visualizzazione statica con layout di Fruchterman-Reingold dei grafi.
5. Visualizzazione interattiva con `PyVis` dei grafi.
6. Calcolo di varie metriche sui grafi.

# 2. Metodologia

## 2.1. Strumenti Utilizzati

Librerie Python

- `requests`: Per interagire con l'API ufficiale di `npm` e raccogliere i dati delle dipendenze in formato JSON.
- `NetworkX`: Per costruire e analizzare i grafi, calcolando le rispettive metriche.
- `PyVis`: Per visualizzare i grafi in modo interattivo.
- `Matplotlib`: Per creare le visualizzazioni statiche dei grafi.
- `numpy`: Per effettuare calcoli e manipolare i dati.

## 2.2. Costruzione dei grafi

### 2.2.1. Grafo diretto

Utilizzando l'algoritmo di visita **BFS** è stato costruito il grafo diretto aciclico (DAG) delle dipendenze in cui i nodi rappresentano i pacchetti e gli archi le relazioni di dipendenza (fino al livello più profondo). Il grafo è stato poi serializzato in formato JSON: `data/grafo_diretto.json`.

### 2.2.2. Grafo indiretto con *preferential attachment*

Successivamente si è costruito un grafo indiretto a partire dal grafo diretto. Esso è stato aumentato utilizzando la tecnica del *preferential attachment*, ovvero aggiungendo nodi artificiali che vengono collegati ai nodi già presenti secondo una probabilità $p$ proporzionale al grado dei secondi. È stato scelto di inserire 400 nodi artificiali con un valore di archi $m$ per nodo

pari a 3 per un totale di 1200 archi. Il grafo risultante è stato poi serializzato in formato JSON: `graphs/grafo_artificiale.json`

#### 2.2.2.1. Grafo indiretto con *random walk*

Con lo stesso numero di nodi (400) e di archi (3) è stata usata la variante del preferential attachment detta *random walk*. Per ogni nodo aggiunto, il primo arco viene connesso a un nodo $K$ scelto uniformemente tra quelli già presenti nel grafo. Per i restanti $m - 1$ archi ogni arco viene connesso con probabilità $p = 0.8$ a uno dei vicini del nodo scelto $K$ (nota se non esiste un vicino disponibile è stato scelto di «perdere» l'arco). In caso contrario, viene scelto un altro nodo uniformemente tra quelli già nel grafo. Il valore di $p = 0.8$ è stato scelto per mantenere un equilibrio tra la località delle connessioni e la casualità. Il grafo risultante è stato poi serializzato in formato JSON: `graphs/grafo_artificiale_v2.json`

# 3. Visualizzazione dei grafi

## 3.1. Statica

Sia per il grafo diretto che per quelli indiretti è stata prodotta una visualizzazione statica utilizzando il layout di Fruchterman-Reingold. Per fare ciò è stata inizialmente creata una figura per la visualizzazione, per poi calcolare le posizioni dei nodi utilizzando il layout Spring e disegnare il grafo sulla figura creata. Le immagini ottenute sono state salvate in `grafici2d/grafico_diretto.png`, `grafici2d/grafico_artificiale.png` e `grafici2d/grafico_artificiale_v2.png`

## 3.2. Interattiva

Dopo aver calcolato i quartili della distribuzione dei gradi, è stata prodotta una visualizzazione interattiva dei grafi utilizzando la libreria `PyVis`.

La dimensione di ciascun nodo è proporzionale al numero dei propri archi in uscita (nei grafi artificiali indiretti *out-degree* = *degree*) e il colore è stato assegnato secondo i seguenti criteri:

- **Grigio**: nodi appartenenti al primo quartile della della distribuzione dei grafi, quindi i nodi con grado minore;
- **Blu**: nodi del secondo quartile;
- **Viola**: nodi del terzo quartile;
- **Giallo**: nodi del quarto quartile;

I grafi prodotti sono stati salvati in formato HTML nei file `html/grafo_diretto_interattivo.html`, `html/grafo_artificiale_interattivo.html` e `html/grafo_artificiale_v2_interattivo.html`.

# 4. Calcolo delle metriche

## 4.1. Grafo diretto

Partendo dal presupposto che il grafo in analisi sia un *DAG*, è noto che esso non sia fortemente connesso. Tuttavia, tale proprietà è stata verificata utilizzando la funzione `nx.is_strongly_connected()`.

A seguito di questa verifica, si è concluso che le misure di centralità globali, come il centro e il raggio, non risultano calcolabili, in quanto richiedono la forte connettività.

Per quanto riguarda la distanza media e la distanza massima, è stato possibile superare il limite imposto dalla non forte connettività calcolando le distanze per ciascun nodo singolarmente e aggregando successivamente i risultati.

Il clustering medio è stato determinato attraverso l'utilizzo della funzione `nx.average_clustering()`, che considera il grafo come indiretto, al fine di garantire coerenza con i grafi artificiali indiretti.

Infine, le metriche *sigma* e *omega* non sono supportate da `NetworkX` per i grafi diretti e, di conseguenza, non è stato possibile calcolarle. Tutte le misure ottenute sono state salvate nel file `misure/misure_grafo_diretto.json`.

## 4.2. Grafi indiretti

Per i due grafi artificiali non diretti, sono state calcolate tutte le misure richieste, ad eccezione del *PageRank*, che non è stato incluso in quanto non significativo per questa tipologia di grafo.

È importante notare che le misure di *in-degree* e *out-degree* sono state equiparate al *degree*, essendo equivalenti in un grafo indiretto.

Per quanto riguarda il calcolo di *sigma* e *omega*, i parametri `niter` e `nrand` della funzione di `nx` sono stati ridotti rispettivamente a 5 e 10. Questa scelta è stata effettuata per ridurre i tempi di esecuzione, mantenendo comunque un'accuratezza accettabile per entrambe le misure.

# 5. Conclusioni

## 5.1. Visualizzazione statica

### 5.1.1. Grafo diretto

Nella visualizzazione ottenuta con lo *spring layout*, i nodi assumono una posizione basata su forze attrattive e repulsive, proporzionali al numero di connessioni. Osserviamo che un *gruppo di nodi centrali* costituisce il «cuore» del grafo, caratterizzato da un alto numero di connessioni: pacchetti altamente dipendenti da altri e pacchetti che fungono da referenti per molti altri, o entrambe le cose. Un numero significativo di *nodi periferici* si dispone lungo la circonferenza. Questi rappresentano pacchetti «base», che non dipendono da altri (grado uscente basso o nullo) e che, allo stesso tempo, hanno pochi pacchetti che dipendono da loro (grado entrante basso).

Questa struttura evidenzia un nucleo complesso di pacchetti fortemente connessi e una periferia composta da librerie semplici e meno integrate nella rete.

### 5.1.2. Grafo indiretto con *preferential attachment*

Nel grafo indiretto generato tramite il modello di *preferential attachment* osserviamo una distribuzione spaziale più «distesa» ma al centro il numero di archi è molto più denso indicando che ci sono dei nodi centrali con un grado molto alto, detti *hub*, che hanno un alto numero di connessioni, coerentemente con il modello del *preferential attachment*, che privilegia i nodi con grado elevato nelle nuove connessioni. I nodi periferici sono distribuiti attorno alla circonferenza e hanno pochi collegamenti, spesso limitati agli hub centrali. Questo sembra suggerire che viene rispettata la distribuzione *power law* attesa che viene confermata anche dal grafico della distribuzione cumulativa dei gradi presente in `grafici2d/power_law.png`

### 5.1.3. Grafo indiretto con *random walk*

Nel grafo indiretto ottenuto con *random walk* osserviamo, a parità di costante $k$ di *spring*, una densità meno alta di archi al centro, ma sopratutto più cluster di nodi e archi tra i nodi periferici. Viene mantenuta comunque una zona centrale con i nodi con più connessioni, gli *hub*, e difatti anche questo grafo conferma oltre che qualitativamente anche nel grafico in `grafici2d/power_law_v2.png` che viene seguita una distribuzione *power law* dei gradi dei nodi.

## 5.2. Visualizzazione Interattiva

La visualizzazione dinamica permette di muovere i nodi, offrendo un'analisi interattiva delle reti. Selezionando un nodo i suoi collegamenti verranno evidenziati, permettendo di visualizzare chiaramente le sue dipendenze.

### 5.2.1. Grafo diretto

Nel grafo diretto notiamo alcuni nodi particolarmente prominenti, come `angular-cli` con grado uscente di 67, indicato dalla sua grande dimensione e colorazione distintiva. Tuttavia, questi nodi sono pochi e si osserva una rapida decrescita nelle dimensioni, con un gran numero di nodi piccoli posizionati lungo la periferia. Questa struttura evidenzia la centralità di pochi pacchetti chiave e la semplicità relativa dei pacchetti periferici, che hanno poche dipendenze o connessioni.

### 5.2.2. Grafo indiretto con *preferential attachment*

Nel grafo indiretto generato con il modello del *Preferential Attachment*, emerge chiaramente la presenza di nodi centrali molto grandi, i cosiddetti *hub*, che hanno un numero estremamente alto di connessioni. Questo risultato è coerente con il meccanismo del modello, che privilegia i nodi con grado maggiore, rafforzando la loro centralità nella rete. Rispetto al grafo diretto, osserviamo un aumento significativo del numero di archi. Inoltre, si nota una maggiore granularità nella dimensione dei nodi: i nodi più piccoli mantengono dimensioni diverse, rappresentando pacchetti con connettività intermedia. Tuttavia, i nodi gialli più grandi sono chiaramente più prominenti rispetto agli altri grafi, indicando la forte influenza del grado nel processo di connessione.

### 5.2.3. Grafo indiretto con *random walk*

Nel grafo indiretto generato con il modello del *Random Walk*, la struttura generale si presenta meno centralizzata rispetto al *Preferential Attachment*. Sebbene si osservino ancora *hub* centrali con un alto numero di connessioni, la loro dimensione è leggermente inferiore rispetto al modello precedente, a causa dell'introduzione di maggiore casualità nel processo di collegamento. Inoltre è ben visibile la presenza di archi tra i nodi periferici a suggerire una maggiore clusterizzazione del grafo. La granularità nella dimensione dei nodi è evidente anche qui, con una distribuzione equilibrata che riflette una gerarchia meno rigida.

## 5.3. Calcolo delle metriche

L'analisi delle metriche sui tre grafi rivela differenze significative nella loro struttura, evidenziando le caratteristiche uniche del grafo diretto e dei due grafi indiretti (*Preferential Attachment* e *Random Walk*).

### 5.3.1. Grafo diretto

Il grafo diretto, essendo un DAG, presenta alcune limitazioni strutturali ma anche caratteristiche distintive. La distanza media è pari a 2.122, la più bassa tra i tre grafi, che riflette una struttura compatta e gerarchica. La distanza massima, invece, è pari a 8, indicando una profondità moderata della rete. Il clustering coefficient, pari a 0.0905, è basso ma non trascurabile, evidenziando una scarsa formazione di triadi. La transitivity, pari a 0.0258, conferma un clustering globale molto basso. Questi risultati mostrano come il grafo diretto tenda a una struttura gerarchica con un clustering limitato.

### 5.3.2. Grafo indiretto con *preferential attachment*

Il grafo indiretto generato con il modello di *Preferential Attachment* mostra una struttura più espansa rispetto al diretto. La distanza media è pari a 4.218, mentre la massima raggiunge 10, evidenziando una rete più ampia e ramificata. Il clustering coefficient è di 0.0705, leggermente inferiore al diretto, segnalando una perdita di clustering nella transizione, mentre la transitivity, pari a 0.0423, suggerisce un leggero aumento di connessioni locali. Il valore di *sigma*, pari a 2.307, conferma che il grafo possiede proprietà «small-world». *Omega*, invece, con un valore di $-0.019$, si avvicina a zero, rappresentando un equilibrio tra struttura randomica e organizzata. Valori inferiori a zero indicano reti più reticolari, mentre valori maggiori segnalano reti casuali. Questi risultati evidenziano come il modello di *Preferential Attachment* bilanci una struttura distribuita con proprietà di rete «small-world».

### 5.3.3. Grafo indiretto con *random walk*

Il grafo indiretto generato con il modello di *Random Walk* evidenzia una struttura più decentralizzata e clusterizzata rispetto al *Preferential Attachment*. La distanza media è pari a 4.919, la più alta tra i tre grafi, riflettendo una struttura meno centralizzata. La distanza massima conferma ciò ed è pari a 12, superiore a quella del *Preferential Attachment*. Il clustering coefficient raggiunge un valore di 0.425, significativamente più alto rispetto agli altri grafi, evidenziando una forte propensione alla formazione di cluster locali. La transitivity è pari a 0.159, la più alta tra i tre grafi, coerente con l'aumento del clustering. Il valore di *sigma*, pari a 11.703, riflette una rete estremamente «small-world», mentre *omega*, con un valore di $-0.144$, indica una rete più decentralizzata rispetto al *Preferential Attachment*, con caratteristiche che tendono maggiormente a un reticolo. Questi risultati mostrano come il modello di *Random Walk* favorisca una struttura più clusterizzata, decentralizzata e **small world**, incrementando significativamente il clustering e la transitivity, ma aumentando anche la distanza media.

## 5.4. Osservazioni conclusive

Il modello di *Preferential Attachment* sacrifica parte del clustering del grafo diretto, con una perdita di 0.02 nel clustering coefficient rispetto al diretto, in favore di una struttura più espansa e di un equilibrio «small-world» quasi ottimale, come indicato da *omega* vicino a $-0.019$. Il modello di *Random Walk*, invece, aumenta notevolmente il clustering, con un incremento di 0.3345 rispetto al diretto, e la transitivity, favorendo la formazione di cluster locali. Questo si riflette anche in un *sigma* più elevato rispetto al *Preferential Attachment*, ma a costo di una distanza media più alta e di un *omega* più negativo. Entrambi i grafi indiretti mostrano distribuzioni *power law*, evidenziando la presenza di hub centrali con gradi elevati, coerenti con i rispettivi modelli generativi.