Alma Mater Studiorum · Università di Bologna

FACOLTÀ DI SCIENZE MATEMATICHE, FISICHE E NATURALI
Corso di Laurea in Matematica

PROPRIETÁ STATISTICHE DELLE CARATTERISTICHE TOPOLOGICHE DEL GRAFO DI UN ″HUB″ GENERATO DA UN SOCIAL NETWORK

Tesi di Laurea in Meccanica Statistica

Relatore:
Chiar.mo Prof.
PIERLUIGI CONTUCCI

Presentata da:
GIOVANNI GASPARE RIGHI

III Sessione
Anno Accademico 2008/2009

a Luisa,
anche se non potrà vedere il mondo
noi glielo racconteremo

Indice

Introduzione

Questa tesi affronta lo studio delle proprietà statistiche della topologia di un grafo, che rappresenta le relazioni inter-personali in un gruppo di utenti di *Facebook*. Perchè è interessante? Quali informazioni produce? Anzitutto va osservato che dalla nascita di internet in poi la globalizzazione ha reso le relazioni sociali un fenomeno di massa con numeri sorprendentemente alti di connessioni. Questo, e la disponibilità dei dati, fornisce una occasione preziosa per analizzarne la struttura.

L'immagine della terra vista dallo spazio, con i confini degli oceani e dei continenti ben definiti, non rappresenta più il mondo interconnesso di oggi. L'evoluzione tecnologica ha fatto sì che i collegamenti tra le diverse parti del globo siano diventati sempre più numerosi e veloci. Da alcuni secoli le navi solcano gli oceani, negli ultimi cinquant'anni gli aerei hanno reso *routine* viaggi che una volta erano avventure lunghissime e venivano riportate nei libri e nei romanzi, quasi fossero leggende. Oggi internet permette di vedere e ascoltare ogni parte del mondo con un semplice *click*.

L'abbattimento dei costi e la facilità di utilizzo, ha reso internet davvero alla portata di tutti e miliardi di persone cercano e scambiano informazioni nella rete virtuale ogni giorno. Quasi sempre i siti utilizzati per lo scambio di dati (*mail, news, instant messaging, ecc ...*) registrano e catalogano questa mole crescente di informazioni, il cui studio diventa fondamentale per comprendere le dinamiche socio-comportamentali dei fruitori, la solidità delle relazioni sociali e molti altri aspetti delle interazioni umane. Forse non si potrà mai prevedere il pensiero di una persona, ma gli studi sui gruppi sociali sono già da tempo diffusi e sono sempre più precisi, basta pensare ai sondaggi preelettorali, che vengono quasi sempre confermati dalle urne.

Facebook, il *social netwok* su cui abbiamo lavorato per estrarre ed analizzare i dati, enuncia già nel suo slogan questa *mission*: ″*Facebook* ti aiuta a mantenere e condividere i contatti con le persone della tua vita.″ A quanto pare, sempre più persone nel mondo sono interessate ad una gestione e manutenzione del loro cosiddetto capitale sociale tramite questo potente strumento, se è vero che il progetto di Mark Zuckenberg, nato nel 2004 con il nome degli annuari scolastici in uso nei college americani, conta nel 2010 400 milioni di utenti attivi. Una delle

peculiarità di questo palcoscenico digitale consiste nel fatto che la costruzione del proprio personaggio (o profilo) non passa attraveverso una elaborazione grafica più o meno sofisticata della pagina web (come ad esempio in *Myspace*): si basa sul caricamento di contenuti di testo/audio/video, ma soprattutto sui contatti (o relazioni) che si hanno. Come dire che gran parte della propria identità virtuale è data dagli amici che si hanno. Proprio per questo motivo, si è dimostrato ad essere particolarmente adatto al nostro tipo di studio l'utilizzo di un profilo di *Facebook*, il social network che più degli altri è stato progettato per favorire la propagazione di informazione attraverso gradi di separazione sociale.

Il profilo prescelto, del quale andremo ad indagare la struttura e che fungerà da *hub*, inteso come grande snodo in cui converge un numero significativo di relazioni a loro volta connesse tra loro, è quello del Circolo Arci Accatà, che lo utilizza per promuovere le sue iniziative sul territorio della Provincia di Bologna e per fidelizzare i suoi soci. Questo punto di collegamento centrale conta 2320 amici (nodi) e 64375 relazioni tra questi.

Per studiare queste interazioni in maniera efficace, la scienza ha creato degli strumenti matematici che schematizzano il problema: i *grafi*, non solo utili per la descrizione visiva del fenomeno, ma anche molto potenti per comprenderne la struttura topologica.

Lo studio dei grafi avviene algebricamente utilizzando le matrici che li rappresentano, dette *matrici delle adiacenze*. I calcoli algebrici sono affidati al calcolatore, che attraverso un programma specifico (in questo caso scritto dall'autore, Appendice B) produce i valori fondamentali per capire la struttura topologica. Soprattutto quando si è in presenza di grandi quantità di dati è importante avere riassunti in pochi valori fondamentali gli oggetti che si stanno analizzando. Nel nostro caso andremo prima a definire e poi a calcolare cosa sono il coefficiente di aggregazione, il grado medio di separazione, la legge di distribuzione dei gradi, per citare solo i valori di riferimento più importanti. Potremo così confrontare questi valori con quelli indicati dai modelli già collaudati di grafi *aleatori* (Erdös e Rény), grafi *small-world* (Watts e Strogatz) e grafi *scale-free* (Barabási e Albert).

Per evitare fenomeni di bordo[1], non utilizzeremo tutti i dati contemporaneamente, ma studieremo intervalli crescenti di porzioni casuali del grafo. In questo modo ricaveremo i dati dei sottografi per un numero di nodi crescente, che ci permetteranno di inferire statisticamente i risultati.

Le funzioni trovate avranno come unico parametro il numero di nodi del grafo. Si potrà quindi calcolare il valore di queste anche per *grafi limite* di dimensione infinita e vedremo come varia la struttura topologica tra grafi di piccole e grandi dimensioni.

[1] Avendo un insieme di dati e scegliendone un sottoinsieme casuale, più la cardinalità del sottoinsieme si avvicina a quella dell'insieme, meno la scelta potrà essere casuale

Andremo così a dimostrare che il nostro *hub* funziona come un *piccolo mondo* e che la distribuzione del numero dei nodi segue la legge della potenza. Da ciò risulta che il tipo di network analizzato è dotato di grande efficenza.

Capitolo 1

Proprietà algebriche e topologiche dei grafi

1.1 I grafi

1.1.1 Definizioni fondamentali

Definizione 1.1 (Grafo)**.** Un *grafo* è un insieme di elementi detti nodi o vertici collegati tra loro da archi o lati. Più formalmente, si dice grafo una coppia ordinata $G = (V, E)$ di insiemi, con V insieme dei nodi ed E insieme degli archi, tali che gli elementi di E siano coppie di elementi di V, ovvero $|E| \leqslant |V|^2$.

Due vertici u, v connessi da un arco e prendono il nome di *estremi dell'arco*; l'arco e viene anche identificato con la coppia formata dai suoi estremi (u, v).

Un arco che ha due estremi coincidenti si dice *cappio*, mentre più archi che connettono gli stessi due estremi danno origine ad un *multiarco*.

Definizione 1.2 (Grafo Semplice)**.** Un grafo sprovvisto di cappi e di multiarchi si dice *grafo semplice*.

Si distinguono inoltre due tipi basilari di grafi, i grafi *orientati* e i grafi *non orientati*. Un grafo orientato D (o *digrafo*, grafo diretto) è un insieme $D = (V, A)$ dove V è l'insieme dei vertici di D e A è l'insieme degli *archi orientati* di D. Un arco orientato è un arco caratterizzato da una *direzione*.

Definizione 1.3 (Grafo non orientato)**.** Un *grafo non orientato* G è un insieme di vertici e archi dove la connessione (u, v) ha lo stesso significato della connessione (v, u).

In questo studio avremo a che fare sempre con *grafi semplici non orientati* e quindi per brevità parleremo semplicemente di *grafi*.

1.1.2 Connettività

Definizione 1.4. Due nodi $u, v \in V$ si dicono *adiacenti* se $\exists\ e = (u, v) \in E$ arco che li contiene.

Definizione 1.5. Il numero di archi incidenti in un vertice $v \in V$ prende il nome *grado* di v.

Un *percorso* o *passeggiata* (in inglese, *walk*) di lunghezza n in G è dato da una sequenza di vertici $v_0, v_1, \cdots, v_n$ (non necessariamente tutti distinti) e da una sequenza di archi che li collegano $(v_0, v_1), (v_1, v_2), \cdots, (v_{n-1}, v_n)$. I vertici v_0 e v_n si dicono *estremi* del percorso.

Un percorso con tutti i nodi distinti prende il nome di *cammino* (in inglese, emphtrail).

Un cammino chiuso ($v_0 = v_n$) si chiama *circuito* o *ciclo*.

Definizione 1.6 (Connessione)**.** Dato un grafo G, due vertici si dicono *connessi* se esiste un percorso con estremi v e u. Se tale percorso non esiste u e v sono detti *sconnessi*.

Osservazione 1. Se il grafo è non orientato la relazione di connessione tra i vertici è una relazione di equivalenza.

Definizione 1.7 (Grafo connesso)**.** Per $i = 1 \dots k$ (k classi di equivalenza) sono definiti i sottografi $G_i = (V_i, E_i)$ come i sottografi massimali che contengono tutti gli elementi connessi tra loro, che prendono il nome di *componenti connesse di* G, la cui cardinalità spesso si indica con $\lambda(G)$.

Se $\lambda(G) = 1$, G si dice *connesso*.

1.1.3 Sottografi

Definizione 1.8 (Sottografo)**.** Il grafo G' è un *sottografo* di G, e si scrive $G' \subseteq G$, se $V' \subseteq V$ e $E' \subseteq E$.

Definizione 1.9 (Sottografo indotto)**.** Se $G' \subseteq G$ e E' contiene tutti gli archi $(u, v) \in E$, per ogni u, $v \in V'$, si dice che G' è un *sottografo indotto* di G.

Se G' è sottografo indotto di G, si dice che esso è *generato* dal suo insieme di vertici V' (o che V' genera G' in G), e si scrive $G' = G[V']$. Pertanto, se $U \subseteq V$ è un qualsiasi sottoinsieme di vertici di G, si indica con $G[U]$ il grafo che ha U come insieme dei vertici e i cui lati sono tutti e soli i lati di G che hanno entrambe le estremità in U. Questi sono i grafi che vengono utilizzati in questo studio: un *set* di vertici viene selezionato a caso tra quelli disponibili, poi si genera il sottografo indotto. Su questo vengono studiate le proprietà del grafo, variando gli elementi

ed il loro numero ad ogni generazione. I dati ricavati dai sottografi di dimensione crescente sono quelli utilizzati per l'inferenza statistica che ci può descrivere il grafo in funzione del numero dei nodi.

1.1.4 Rappresentazione

Ci sono due modi per rappresentare un grafo con una struttura dati utilizzabile da un programma: la *lista delle adiacenze* e la *matrice delle adiacenze*. In una lista di adiacenze, una lista mantiene un elenco di nodi, ed ad ogni nodo è collegata una lista di puntatori ai nodi collegati da un arco. In una matrice di adiacenze, una matrice N per N dove N è il numero dei nodi, mantiene un valore *vero*$(= 1)$ in una cella (u, v) se il nodo u è collegato al nodo v.

Indicati con N la cardinalità dell'insieme dei nodi e con E la cardinalità dell'insieme degli archi del grafo, le due rappresentazioni, quella mediante liste e quella mediante matrice delle adiacenze, richiedono rispettivamente $O(N + 2E)$ e $O(N^2)$ spazio di memoria.

Ognuna delle due rappresentazioni ha alcuni vantaggi: una lista delle adiacenze risulta più adatta a rappresentare grafi sparsi o con pochi archi, perciò è facile trovare tutti i nodi adiacenti ad un nodo dato; una matrice delle adiacenze è invece più indicata per descrivere grafi densi con molti archi (inoltre è più facile implementare algoritmi, che si possono basare sulle proprietà algebriche delle matrici).

Nel seguito approfondiremo soltanto le matrici di adiacenza, che sono state utilizzate per implementare il programma di calcolo.

1.2 Matrici di adiacenza

Definizione 1.10 (Matrice di adiacenza)**.** Dato un grafo $G = (V, E)$ la sua matrice di adiacenza $A \in \mathbb{M}_{|V|\times|V|}$ è costituita da una matrice binaria quadrata che ha come indici di righe e colonne i nomi dei vertici del grafo. Nel posto (i, j) della matrice si trova 1 se e solo se esiste nel grafo un arco che va dal vertice i al vertice j, altrimenti si trova 0.

$$A = (a_{i,j}) = \begin{cases} 1 & \text{se } (i,j) \in E \\ 0 & \text{altrimenti} \end{cases}$$

Osservazione 2. Se al posto dei soli valori binari 1 e 0 si trovano anche altri numeri, questi sono da interpretare come il *peso* attribuito a ciascun arco. In questo caso però le proprietà algebriche seguenti non sono più valide. Nel nostro studio useremo matrici di adiacenza con valori binari.

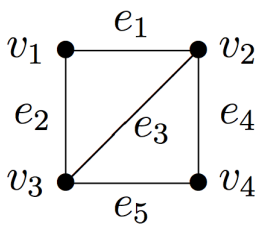


Figura 1.1: Esempio di un grafo semplice

$$A = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix}$$

Figura 1.2: Matrice delle adiacenze del grafo Fig: 1.1

Osservazione 3. Nel caso della rappresentazione di grafi *non orientati*, come ad esempio il nostro, la matrice è *simmetrica* rispetto alla diagonale principale. Inoltre, nelle matrici di adiacenza dei grafi *semplici* (o comunque privi di cappi), il valore $a_{i,i}$ sulla diagonale è sempre 0.

Osservazione 4. Una delle caratteristiche fondamentali della matrice di adiacenza è di permettere di ottenere il numero delle *passeggiate* da un nodo i ad un nodo j che devono attraversare n nodi. Per ottenere tutto ciò è sufficiente la potenza $n-esima$ della matrice e vedere il numero che compare al posto (i, j). Questo risultato è dato dal seguente

Teorema 1.2.1. *Sia A la matrice di adiacenza di un grafo $G = (V, E)$, con*

$$V = \{v_1, v_2, \cdots, v_n\}.$$

Allora, per ogni $k \geqslant 1$, l'elemento di posto (i, j) della matrice A^k è il numero delle passeggiate di lunghezza k che hanno come estrmità i vertici v_i e v_j.

Dimostrazione. La dimostrazione si fa per induzione su k. Per $k = 1$ l'enunciato è banalmente vero: esiste una v_i-v_j passeggiata di lunghezza 1 se e solo se i vertici

v_i e v_j sono adiacenti, cioè se e solo se l'elemento di posto (i, j) della matrice A è uguale a 1.

Sia quindi $k > 1$ e supponiamo che l'enunciato valga per $k - 1$. Poniamo $A^{k-1} = (a_{i,j}^{(k-1)})$. In base all'ipotesi induttiva, $a_{i,j}^{(k-1)}$ è il numero di v_i-v_j passeggiate di lunghezza $k - 1$ in G. Se poniamo $A^k = (a_{i,j}^{(k)})$, dall'uguaglianza $A^k = A^{k-1} \cdot A$ discende che

$$a_{i,j}^k = \sum_{l=1}^{n} a_{i,l}^{(k-1)} \; a_{l,j} \; .$$

Osserviamo che ogni v_i-v_j passeggiata di lunghezza k consiste in una v_i-v_l passeggiata di lunghezza $k - 1$ seguita dal lato (v_l, v_j), per un qualche vertice v_l adiacente a v_j. Possiamo quindi affermare che il numero di v_i-v_j passeggiate di lunghezza k è la somma dei numeri delle v_i-v_l passeggiate di lunghezza $k - 1$, ove la somma è estesa a tutti i vertici v_l adiacenti a v_j (cioè tali che $a_{l,j} = 1$). Dalla formula precedente, ricordando che (per ipotesi induttiva) $a_{i,l}^{(k-1)}$ è proprio il numero di v_i-v_l passeggiate di lunghezza $k - 1$, si conclude quindi che $a_{i,j}^{(k)}$ coincide con il numero di v_i-v_j passeggiate di lunghezza k. □

1.3 Valori fondamentali di un grafo

1.3.1 Grado dei vertici

Definizione 1.11. Sia $G = (V, E)$ in grafo. Il *grado* (o *valenza*) di un vertice $v \in V$ è il numero di lati incidenti a v. Esso è indicato con $k(v)$ o k_v. Con $< k >$ si indica poi il *grado medio* dei vertici di G, che si ottiene facendo la media dei valori $k_v \; \forall v \in V$

Osservazione 5 (Significato di A^2). Abbiamo visto sopra il significato di A^k, nel caso $k = 2$ abbiamo che il valore di $c_{i,j}$ della matrice A^2 corrisponde al numero di nodi che incidono sia il vertice i sia il vetrice j. Allora per $i = j$, cioè sulla diagonale di A^2, si avrà il valore k_i, cioè il grado del vertice i.

Un altro modo per calcolare k_i, data la matrice delle adiacenze, è sommare i valori della colonna (o riga, ricordiamo che la matrice A è simmetrica per grafi semplici indiretti) i-esima.

$$k_i = \sum_{j=1}^{n} a_{i,j} = \sum_{j=1}^{n} a_{j,i} = \sum_{j=1}^{n} a_{i,j} \; a_{j,i} = c_{i,i} \in diag(A^2)$$

1.3.2 Legge di distribuzione dei gradi

Definizione 1.12. Un grafo è detto *regolare* se tutti i suoi vertici hanno lo stesso grado. Se tale grado è pari a k, si dice anche che il grafo è k-regolare.

Spesso non tutti i vertici di un grafo hanno lo stesso numero di archi incidenti. Il valore del grado dei nodi è caratterizzato dalla funzione di distribuzione $P(k)$, che esprime la probabilità che un nodo selezionato a caso abbia esattamente k archi. Si vedrà più avanti l'importanza centrale della funzione $P(k)$, infatti questa legge definisce la natura e l'evoluzione del grafo.

1.3.3 Gradi di separazione

Definizione 1.13. La *distanza* di due vertici u e v in un grafo G, indicata con $l(u, v)$, è la lunghezza del più corto $u - v$ cammino in G (cioè una *geodetica*). Se un tale cammino non esiste, si pone $l(u, v) = +\infty$, ciò accade se u e v non appartengono alla stessa componente connessa del grafo.

Ripetendo il calcolo per tutte le possibili coppie di vertici e facendo la media dei valori ottenuti (omettendo i valori infiniti) si ottiene il *grado di separazione* del grafo, cioè il numero medio di passi che occorrono per connettere due nodi del grafo (ovviamente appartenenti alla stessa componente connessa). Il grado di separazione si indica con l oppure con $< l >$.

Data la matrice delle adiacenze di un grafo G, esistono algoritmi che calcolano la distanza minima di due nodi. Il risultato dell'algoritmo è a sua volta una matrice D dove nella posizione (i, j) si ha il valore della distanza tra i nodi i e j. In questa analisi è stato utilizzato l'algoritmo Flooyd - Warshall (Appendice A).

1.3.4 Coefficiente di aggregazione

Nella teoria dei grafi, il *coefficiente di aggregazione* misura quanto i nodi vicini tendono ad essere legati tra loro (aggregati). Diverse prove hanno suggerito che in gran parte delle reti reali, e in particolare nei social network, i nodi tendono a creare dei gruppi caratterizzati da una densità relativamente alta di collegamenti (Watts e Strogatz) [1]. Questo fenomeno, che si trova spesso nei network reali, è invece meno evidente nei grafi implementati casualmente (vedi Cap. 2.1).

Definizione 1.14. Il *coefficiente di aggregazione* di un vertice in un grafo definisce quanto i suoi vicini tendono a creare un grafo completo. Chiamiamo N_i l'insieme dei nodi collegati al vertice v_i, cioè

$$N_i = \{v_j \mid (v_i, v_j) \in E\}$$

ovviamente la cardinalità di N_i è data dal grado k_i del vertice v_i. Il coefficiente di aggregazione C_i del vertice v_i è data dalla proporzione tra il numero di collegamenti tra i nodi a lui vicini (cioè in N_i) e il numero di collegamenti massimo che potrebbero esistere tra gli stessi. Chiamiamo

$$E_i = |\{(v_j, v_k)\}| : v_j, v_k \in N_i \ ; \ (v_j, v_k) \in E$$

inoltre, se un vertice ha k_i vicini, il numero massimo di collegamenti tra di essi è dato da $k_i(k_i - 1)/2$. Allora

$$C_i = \frac{2\ E_i}{k_i(k_i - 1)} \ .$$

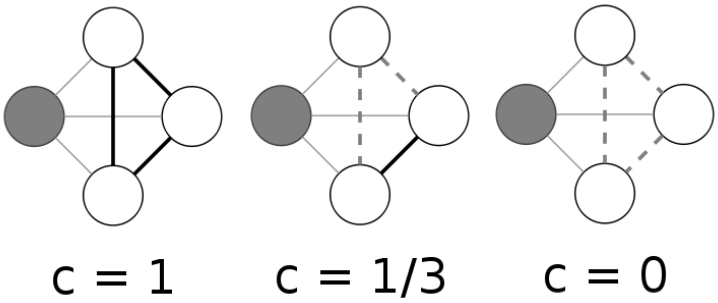


Figura 1.3: Coefficiente di aggregazione calcolato per il nodo scuro. I segmenti tratteggiati indicano la mancanza del collegamento

Proposizione 1.3.1 (Calcolo del coefficiente di aggregazione)**.** *Il coefficiente di aggregazione può essere calcolato a partire dalla matrice delle adiacenze. Infatti sulla diagonale di A^3 si trovano i valori di $2E_i$.*

Dimostrazione. Se esiste un arco che collega due vertici j e k che appartengono a N_i, equivale a dire che esiste un ciclo di lunghezza 3 dei nodi i, j, k. La matrice A^3 nella posizione (i, j) descrive quanti percorsi di lunghezza 3 esistono tra il vertice i ed il vertice j. Se $i = j$ la matrice dice quanti sono i cicli di lunghezza 3 (anche detti *triangoli*) di cui il vertice i fa parte. Visto che il grafo è indiretto lo stesso triangolo può essere percorso in entrambe le direzioni, quindi verrà contato due volte, da qui segue che $diag(A^3) \ni d_i = 2E_i$.

□

Il coefficiente di aggregazione di un grafo G è la media dei coefficienti C_i dei singoli vertici:

$$C = \frac{1}{n} \sum_{i=1}^{n} C_i$$

Capitolo 2

Modelli fondamentali

Gli studiosi basandosi sulle proprietà dei grafi elencate sopra hanno creato dei modelli che permettono di indagare i network reali. I modelli sono utili fondamentalmente per due motivi: permettono di creare un grafo con determinate caratteristiche (in particolare la dimensione) molto velocemente ed ogni volta che sia necessario, mentre l'attività di raccolta dei dati, quando si tratta di network reali, è un procedimento lungo che oltretutto restituisce un solo grafo. Quello che si può fare, e che faremo nel nostro studio è, dato un *set* di dati reali, confrontare porzioni di grafo (quindi sottografi indotti di dimensione variabile) e verificare se mantengono certe proprietà che caratterizzano i modelli per i grafi complessi.

Di seguito analizzeremo brevemente i tre modelli fondamentali di network.

2.1 Grafo Aleatorio

Il modello *Aleatorio*, in inglese *Random*, è stato introdotto da *Paul Erdös* e *Alfréd Rényi* (1959-1961) [3]. I nodi del grafo sono collegati casualmente con una probabilità fissata $p \in [0, 1]$. Questo modello probabilistico è stato spesso usato per affrontare reti complesse molto estese. É anche stato il primo modello in cui è intervenuta la statistica, che in seguito si è poi rivelata fondamentale per trattare i problemi della teoria dei grafi.

Data un rete G=(V,E), definiamo la cardinalità dei due insiemi V ed E, $N = |V|$ e $n = |E|$, si può calcolare p dividendo il numero degli archi di G per il numero di archi possibili:

$$p = \frac{2n}{N(N-1)}$$

questo valore esprime proprio la *densità* del grafo. Vediamo anche come calcolare il grado medio di un grafo Random: un nodo sarà collegato in media ad una

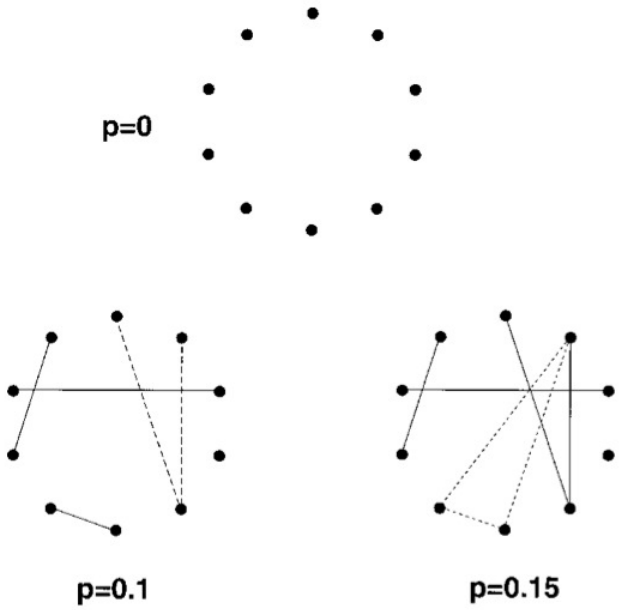


Figura 2.1: Processo di evoluzione del modello Random. All'inizio ci sono $N = 10$ nodi isolati ($p = 0$), poi ogni coppia di vertici è connessa con probabilità p nelle figure in basso. Per $p = 0.15$ la metà dei nodi fanno già parte di un'unica componente connessa.

percentuale p dei nodi presenti nel grafo:

$$< k >= \frac{2n}{N} = p\ (N-1) \simeq p\ N.$$

Erdös e Rényi sono stati i primi a studiare la distribuzione del massimo e minimo grado in un grafo random. La legge di distribuzione dei gradi arriverà invece più tardi, nel 1981 ad opera di *Bollobás* [5].

In un grafo Random con una densità p il grado k_i del nodo i segue la distribuzione binomiale con parametri $N-1$ e p:

$$P(k_i = k) = C_{N-1}^{k} p^k (1-p)^{N-1-K} \tag{2.1}$$

Questa probabilità rappresenta il numero di modi in cui k archi possono essere tracciati da un certo nodo: la probabilità di k archi è p^k, la probabilità che non ci siano altri archi è $(1-p)^{N-1-k}$, e infine ci sono C_{N-1}^{k} modi equivalenti di selezionare i k nodi di arrivo per gli archi. Per trovare la legge di distribuzione dei gradi dobbiamo studiare il numero di nodi con lo stesso grado k, X_k. Dobbiamo determinare la probabilità che X_k abbia un valore dato, $P(X_k = r)$.

Dalla 2.1, il valore atteso per per il numero di nodi con grado k è

$$E(X_k) = NP(k_i = k) = \gamma_k \tag{2.2}$$

dove

$$\gamma_k = NC^k_{N-1}p^k(1-p)^{N-1-K} \tag{2.3}$$

La distribuzione dei valori di X_k, $P(X_k = r)$, è approssimata dalla distribuzione di Poisson

$$P(X_k = r) = e^{-\gamma_k}\frac{\gamma_k^r}{r!} \tag{2.4}$$

Quindi il numero di nodi con grado k segue una distribuzione di Poisson con valore medio γ_k. La distribuzione di Poisson decresce rapidamente per grandi valori di r. Con un po' di semplificazione possiamo dire che 2.4 implica che X_k non diverga molto dal suo valore atteso 2.2. Quindi possiamo dire che la legge di distribuzione dei gradi per il grafo Random è binomiale,

$$P(k) = C^k_{N-1}p^k(1-p)^{N-1-K} \tag{2.5}$$

che per N grande è approssimata dalla distribuzione di Poisson

$$P(k) \simeq e^{-pN}\frac{(pN)^k}{k!} = e^{-<k>}\frac{<k>^k}{k!} \tag{2.6}$$

I network reali in generale hanno coefficiente di aggregazione abbastanza alto rispetto a quello che ci si aspetta in un modello random. Se consideriamo un nodo di un grafo di questo tipo e tutti i suoi nodi adiacenti, la probabilità che due di questi vicini siano connessi equivale alla probabilità che due nodi selezionati a caso siano connessi. Allora il coefficiente di aggregazione di una grafo Random è dato da

$$C_{rand} = p = \frac{<k>}{N} \tag{2.7}$$

Vediamo velocemente altri due aspetti simili della connessione nelle reti Random: il *diametro* e il *grado di separazione*. Il diametro di un grafo è la massima distanza tra due punti qualsiasi. Si avrebbe quindi che il diametro di un grafo disconnesso è infinito, ma può essere definito come il massimo diametro delle sue componenti connesse (in inglese *cluster*). I grafi Random tendono ad avere un diametro piccolo. La ragione di ciò è che è molto probabile che un network di questo tipo sia *sparso* (poco aggregato), allora molto probabilmente il numero di nodi a distanza l da un vertice dato non è molto più piccola di $<k>^l$. Uguagliando $<k>^l$ a N e ricavando l troviamo la distanza massima a cui mediamente due nodi si possono trovare e quindi stiamo valutando il diametro d del grafo. Vediamo

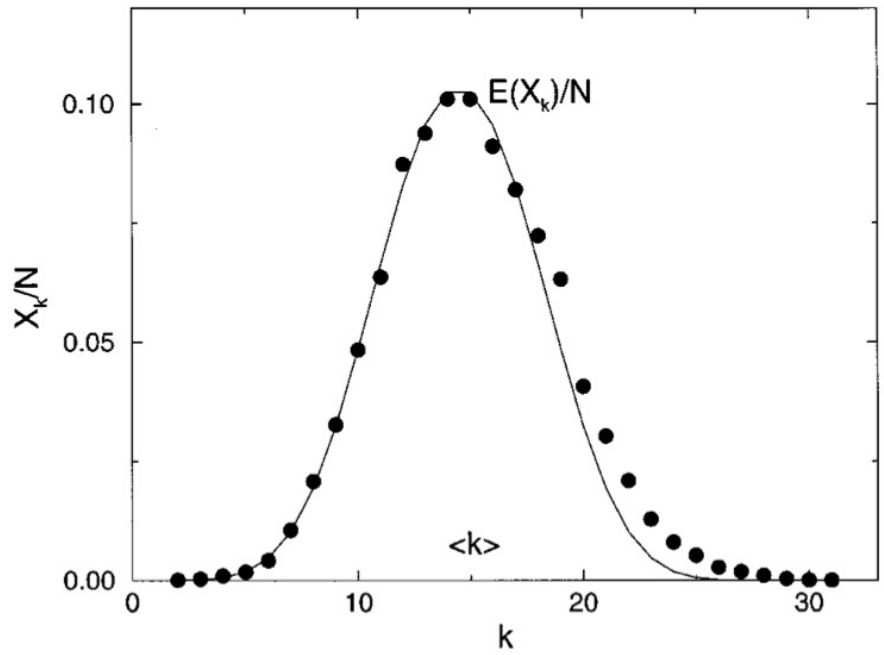


Figura 2.2: Legge di distribuzione dei gradi in un grafo Random che risulta da una simulazione numerica. É stato generato un singolo grafo con $N = 10000$ nodi probabilità di connessione $p = 0,0015$, poi è stato calcolato il numero di nodi con grado k. Il grafico confronta X_k/N con la distribuzione di Poisson, $E(X_k)/N = P(k_i = k)$. Si può vedere che la differenza è minima.

quindi che il diametro è proporzionale a $\ln(N)/\ln(< k >)$, e quindi dipende solo logaritmicamente dal numero di nodi.

Il diametro di un grafo Random è stato studiato da molti autori (si veda ad esempio Chung e Lu[1]). La conclusione generale è stata che per la maggior parte dei valori di p, quasi tutti i grafi con lo stesso N e la la stessa p hanno praticamente lo stesso diametro. Cioè l'intervallo di valori in cui il diametro di queste reti varia è molto piccolo e concentrato attorno a

$$d = \frac{\ln(N)}{\ln(pN)} = \frac{\ln(N)}{\ln(< k >)} \tag{2.8}$$

Per quando visto sopra ci si aspetta che anche il grado di separazione, cioè la distanza media di tutte le coppie di vertici del grafo, come il diametro segua un andamento logaritmico rispetto al numero di nodi:

[1]Chung F. e L. Lu, 2001, Adv. Appl. Math. 26,257.

$$l_{rand} \simeq \frac{\ln(N)}{\ln(< k >)} \tag{2.9}$$

Possiamo quindi concludere che un network generato casualmente sarà molto sparso, visto che non ci sono collegamenti preferenziali tra i nodi, e tra questi sarà quasi impossibile trovare gradi k_i molto più alti del grado medio $< k >$. Inoltre Erdös ha dimostrato che la rete casuale è la più efficiente per collegare quasi del tutto un insieme di punti isolati, cioè basta una piccola percentuale delle possibili connessioni tra i nodi; inoltre ha stabilito che questa percentuale diminuisce all'aumentare della dimensione della rete. Ad esempio in una rete di 300 punti vi sono quasi 50.000 connessioni possibili, ma ne bastano il 2% per garantire il collegamento completo. Nel caso di 1000 punti la percentuale necessaria scende all'1% e così via [1].

2.2 Small-World

L'idea chiave di questo modello *Piccolo Mondo* (dall'inglese *Small World*) proposto da Duncan Watts e Steven Strogatz nel 1998 [2], è di prendere un grafo ordinato (un *reticolo*, in inglese *lattice*) di dimensione finita e aggiungere o modificare al suo interno qualche collegamento tra i nodi casualmente. In questo modo il coefficiente di aggregazione abbastanza alto che si trova nei network reali viene rispecchiato dal modello, cosa che invece non accade nei modelli Random. Vediamo più nel dettaglio l'algoritmo del modello (vedi Fig 2.3):

1. *Ordine*: iniziamo con un reticolo ad anello con N nodi in cui ogni nodo è connesso con i suoi primi K vicini ($K/2$ su ogni lato). Per avere un grafo poco denso ma connesso consideriamo $N \gg K \gg \ln(N) \gg 1$.

2. *Un po' di disordine*: ricollegare a caso ogni nodo del reticolo con probabilità p, escludendo cappi e multiarchi. Questo processo introduce nel grafo $pNK/2$ archi *a lungo raggio* che connettono nodi che altrimenti avrebbero fatto parte di aggregazioni diverse. Variando p si vede bene la transizione tra l'ordine ($p = 0$) e l'irregolarità ($p = 1$).

Questo modello ha origine dagli studi dei collegamenti sociali, nei quali quasi tutte le persone fanno parte di gruppi di amici in cui la conoscenza reciproca è condivisa (i colleghi, gli abitanti di una strada, ecc...) e allo stesso tempo ognuno ha qualche amico lontano (persone in altri paesi o amicizie d'infanzia) che rappresentano gli archi a lungo raggio del modello Piccolo Mondo.

In questo modello coesistono un basso grado di separazione l e un alto coefficiente di aggregazione C. Per capire il comportamento di questi due parametri

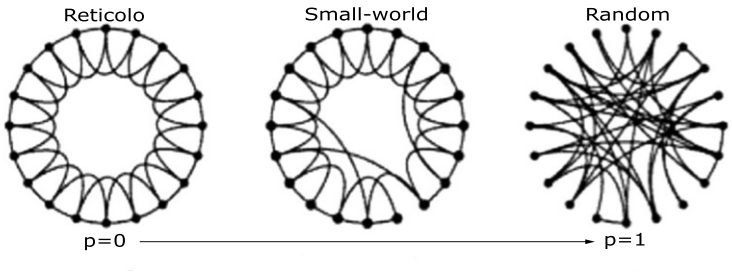


Figura 2.3: Costruzione di un grafo Small-World senza alterare il numero di nodi e archi. $N = 20$ $K = 4$, per $p = 0$ l'anello iniziale rimane invariato, al crescere di p cresce il disordine fino ad arrivare in $p = 1$ ad un grafo con tutte le connessioni completamente casuali

dobbiamo considerarli in funzione del parametro del modello p. Per il reticolo considerato prima ($p = 0$) $l(0) \simeq N/2K \gg 1$ e $C(0) \simeq 3/4$; quindi l aumenta linearmente con la dimensione del sistema e il coefficiente di aggregazione è alto. Al lato opposto, $p \to 1$, il modello converge ad un grafo Random per cui $l(1) \sim \ln(N)/\ln(K)$ e $C(1) \sim K/N$, quindi l cresce come il logaritmo di N e C è inversamente proporzionale ad N. Questi casi limite potrebbero suggerire che un coefficiente C alto sia sempre legato ad un alto l ed un piccolo C associato ad un piccolo l. Al contrario Watts e Strogaz trovarono un ampio intervallo di p in cui $l(p)$ è vicino ad $l(1)$ e allo stesso tempo $C(p) \gg C(1)$. Si ha quindi una rapida discesa di $l(p)$ già per piccoli valori di p, al contrario $C(p)$ rimane pressochè invariato, ne risultano così grafi aggregati con un piccolo grado di separazione (vedi Figura 2.4). Queste due caratteristiche si riscontrano proprio nei network reali, il che li ha portati ad essere chiamati *piccoli mondi*, anche se poi differiscono da questo modello per altre importanti caratteristiche, come la legge di distribuzione dei gradi [3].

Non ci soffermiamo molto sulla legge di distribuzione dei gradi di questo modello (che si può approfondire su [4, 3]); i primi studi pubblicati sono di Barrat e Weigt (2000) e dimostrano che se $p \neq 0$ (altrimenti si ha che tutti nodi hanno grado K) l'andamento della legge di distribuzione è simile a quella dei grafi Random. Si ha un picco in corrispondenza di $< k >= K$ e un decadimento esponenziale per k grande. Quindi la topologia del grafo è abbastanza omogenea, tutti i nodi hanno più o meno lo stesso grado.

2.3 Scale-Free

Una delle scoperte più importanti nello studio delle reti complesse è stata che nella maggior parte dei network (soprattutto in quelli più grossi) la legge di di-

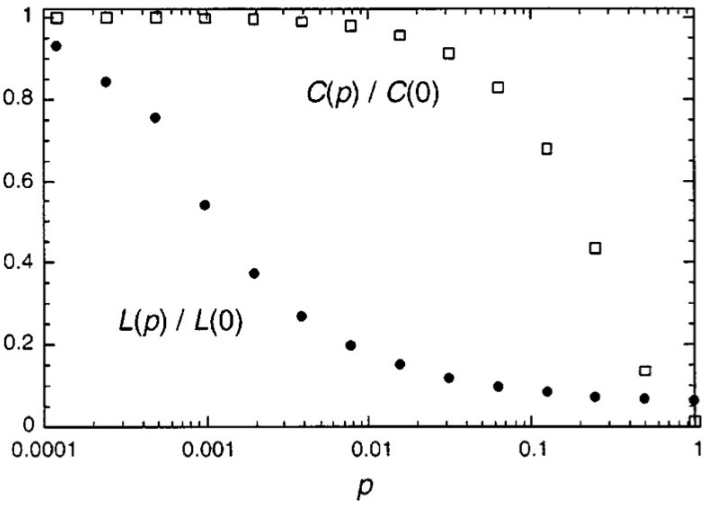


Figura 2.4: Grado di separazione $l(p)$ e coefficiente di aggregazione $C(p)$ per il modello piccolo mondo. I dati sono normalizzati dai valori $l(0)$ e $C(0)$ del reticolo regolare. La scala logaritmica orizzontale fa vedere la rapida discesa di $l(p)$, mentre $C(p)$ rimane praticamente costante. Questo è l'intervallo che corrisponde al fenomeno piccolo-mondo.

stribuzione dei gradi si discosta molto dalla distribuzione di Poisson (valida per i modelli citati sopra). In particolare molti network persentano nella *coda* (per valori altri di k) una distribuzione che segue la *legge della potenza*

$$P(k) \sim k^{-\gamma}. \tag{2.10}$$

Queste reti sono chiamate *scale-free* (Barabási e Albert, 1999) [6].

Definizione 2.1 (Distribuzione ad invarianza di scala)**.** Si dice che una distribuzione di probabilità $P(x)$ è a *scala* n-*invariante* se il suo momento semplice n-esimo $\mu_n = +\infty$, cioè:

$$\mu_n = \int_{-\infty}^{+\infty} x^n P(x) dx = +\infty$$

Si vede subito che se $P(k) \sim k^{-\gamma}$ e $0 < \gamma < +\infty$ sicuramente esiste n tale che l'integrale del momento semplice n-esimo diverge.

La distribuzione dei gradi secondo la legge della potenza osservata in molti network reali è stata la base su cui Barabási e Albert formularono il loro modello, aggiungendo due aspetti fondamentali che prima non erano ancora stati considerati. Primo, mentre i due modelli discussi sopra iniziano da un numero fissato N di

vertici, per poi collegarli o ricollegarli casualmente, senza modificare N, al contrario la maggior parte dei network del mondo reale descrivono sistemi che *crescono* o mutano grazie alla continua aggiunta di nuovi nodi. Partendo da un piccolo nucleo di vertici, il loro numero aumenta durante la vita della rete. Ad esempio, il *World Wide Web* cresce esponenzialmente nel tempo grazie all'aggiunta di nuove pagine *web*.

Secondo, finora i modelli discussi assumono che la probabilità con cui due nodi si connettono è indipendente dal loro grado. In realtà le reti reali mostrano che esistono *collegamenti preferenziali*, come il fatto che spesso la probabilità di connessione ad un nodo dipende dal suo grado (cioè quanti collegamenti ha già). Ad esempio, una nuova pagina *web* includerà molto più probabilmente *link* a pagine già popolari, e quindi già ricche di collegamenti.

Questi due ingredienti, crescita e collegamenti preferenziali, hanno ispirato il modello di Barabási e Albert, che per primo ha generato un network che segue la legge della potenza nella distribuzione dei gradi. Vediamo brevemente l'algoritmo:

1. *Crescita*: partendo da un piccolo numero (m_0) di nodi, ad ogni passaggio (*step*), viene aggiunto un vertice con $m(\leqslant m_0)$ archi che collegano il nuovo vertice ad m nodi già presenti nel sistema.

2. *Collegamento o aggregazione preferenziale*: Quando vengono scelti i nodi a cui il nuovo vertice si connette, viene assunto che la probabilità Π che un nuovo nodo sia collegato al nodo i dipende dal grado k_i del nodo i, cioè:

$$\Pi(k_i) = \frac{k_i}{\sum_j k_j}$$

Dopo t passaggi questo algoritmo crea un grafo con $N = t + m_0$ vertici e mt archi. Simulazioni numeriche hanno indicato che questo network evolve in un grafo ad invarianza di scala; la probabilità che un nodo abbia k archi segue la legge della potenza con esponente $\gamma_{BA} = 3$ [3].

La legge della potenza appena descritta implica che i nodi con grado molto elevato non siano una quantità marginale. Questi nodi sono detti *hub* e sono quelli che tengono connessa la rete. Durante la crescita del grafo, per l'aggregazione preferenziale, questi nodi aumentano molto più velocemente degli altri il loro grado, autoalimentando questo fenomeno detto del *ricco sempre più ricco*. Questo fatto descritto inizialmente in economia, grazie agli studi sulle connessioni dei network è stato riscontrato anche in molti altri ambiti di ricerca, ad esempio nei collegamenti tra le citazioni degli articoli scientifici o nelle interazioni delle proteine.

Abbiamo visto quindi che il modello Scale-free fotografa abbastanza bene il comportamento della coda per le distribuzioni che seguono legge della potenza.

Vediamo adesso se le altre proprietà del modello rispecchiano la realtà: una caratteristica fondamentale dei piccoli mondi è la coesistenza di aggregazione e vicinanza (un valore basso del grado di separazione medio).

Per ricavare C ed l (al solito, rispettivamente, coefficente di aggregazione e grado di separazione) è stata utilizzata l'inferenza di simulazioni numeriche e le funzioni dipendono da N (che aumenta progressivamente). Il grado di separazione nel modello Barabási-Albert è più piccolo di quello di un grafo Random per qualsiasi N, indicando che la topologia eterogenea della rete scale-free è più efficiente nel tenere i nodi vicini rispetto alla topologia omogenea dei grafi random. Si è visto inoltre che l cresce proporzionalmente al logaritmo di N:

$$l \sim \ln(N).$$

Non esiste una predizione analitica neanche per il coefficiente di aggregazione C, ma i risultati numerici per differenti grandezze del network sono stati comparati con un grafo random con gli stessi parametri N e $< k >$. Si è trovato che il coefficiente di aggregazione di un grafo scale-free è circa cinque volte più alto di quello di un grafo random, e questo fattore aumenta leggermente con il numero dei nodi. Rimane comunque il fatto che il coefficiente C nel modello Barabási-Albert diminuisce con con la crescita del network, seguendo approssimativamente la legge della potenza $C \sim N^{-0,75}$, che, mentre è una decrescita più lenta rispetto a $C_{rand} =< k > N^{-1}$, è un comportamento diverso da quanto visto nel modello small-world, dove C è indipendente da N.

Capitolo 3

Studio dell'hub di un social network

3.1 I social network

Una rete sociale (in inglese *social network*) consiste di un qualsiasi gruppo di persone connesse tra loro da diversi legami sociali, che vanno dalla conoscenza casuale, ai rapporti di lavoro, ai vincoli familiari. Grazie anche alla rapida evoluzione in atto nel settore delle nuove tecnologie e della comunicazione, nell'epoca del *web 2.0* una parte sempre più significativa della vita sociale si svolge su internet. Qui proliferano portali liberamente accessibili che fungono da aggregatori (o *social network*) nei quali gli utenti registrati si possono incontrare, stringere amicizie virtuali, scambiare informazioni o materiali multimediali.

Questi social network (qui analizzeremo nello specifico *Facebook*), si possono matematicizzare in maniera naturale come grafi: i profili degli utenti sono i nodi, le relazioni tra di essi (nel caso di Facebook le amicizie) sono gli archi che li collegano. Conoscere la topologia di un intero social network non ci è possibile, in quanto i dati a cui bisognerebbe accedere sono altamente sensibili e di conseguenza molto preziosi e ben protetti. Facebook ad esempio, permette di visualizzare completamente soltanto i profili di quei membri con cui già si è stretta amicizia. É quindi possibile arrivare a conoscere quali sono gli *amici degli amici* (dall'inglese *Friends of a friends* o FOAF). Per il nostro studio abbiamo utilizzato un profilo di Facebook (che chiamiamo *Root*) con 2320 relazioni. Incrociando questi dati, alcuni programmi tra cui *FOAF Exporter* estrapolano le relazioni di amicizia di secondo livello. Ad esempio, se Marco e Anna hanno entrambi come amico Flavio e tutti e tre sono amici del profilo Root, il programma restituisce le coppie Marco-Flavio e Anna-Flavio. Questo sottografo del social network rappresenta il nostro insieme di dati di partenza.

Se considerassimo nello studio del grafo anche le connessioni dipendenti direttamente dal Root, lo studio apparirebbe poco interessante, perchè tale dipendenza sbilancerebbe il grafo. Ad esempio, la distanza tra due nodi qualsiasi sarebbe sempre al massimo due. Quindi tralasceremo volontariamente le informazioni relative all'*hub* di partenza e ci concentreremo solamente sulle relazioni che sussistono tra i nodi ad esso collegati. Il grafo che ne deriva, prenderà il nome di G.

3.2 Analisi dei dati

Basandoci sulle definizioni riportate nel Cap.1, abbiamo costruito un programma utilizzando il linguaggio C (Appendice B), per studiare i dati caratteristici di sottografi indotti di G. Come anticipato nell'Introduzione, non studiamo G nella sua interezza, per evitare i fenomeni di bordo. Dato N minore del numero di nodi di G, vengono quindi selezionati casualmente N nodi di G e da questi viene costruito il sottografo indotto G_n. Per ogni G_N vengono calcolati i valori del grado della distribuzione, il numero di amici medio e tutti i valori caratteristici del sottografo. Il valore di N varia tra 100 e 1000 ad intervalli regolari. Contemporaneamente, per N fissato, viene generato più di un sottografo, selezionando ogni volta gli N nodi di partenza, tra quelli di G.

3.2.1 Studio di un sottorgafo di dimensione 1000

Vediamo ora i risultati dello studio di un singolo sottografo $G_N = (V_N, E_N)$ di G, con $N = 1000$ vertici selezionati a caso.

Dopo aver memorizzato i nomi dei nodi, si compila la matrice di adiacenza $A = (a_{i,j})$, di dimensione $N \times N$, vagliando le 64375 relazioni del grafo G. Quelle che coinvolgono una coppia di nodi (i, j) *t.c.* $i, j \in V_N$ vengono memorizzate in A, ed essendo il grafo indiretto, si ha che $a_{i,j} = a_{j,i} = 1$. Subito si può calcolare il grado di ogni vertice, abbiamo visto infatti che se k_i rappresenta il grado del vertice i

$$k_i = \sum_{j=1}^{N} a_{i,j} \ ,$$

e successivamente possiamo contare le ripetizioni dei valori k_i. Normalizzando tali valori al numero totale di nodi N, si ha la probabilità $p(k)$ che un vertice abbia grado k nel grafo G_N.

Vediamo una porzione dei dati nella tabella 3.1.

Rappresentando questi dati in un grafico (Figura 3.1) con doppia scala logaritmica si vede che i dati seguono un andamento abbastanza lineare appena k cresce.

k	*ripetizioni*	$p(k) = ripetizioni/N$
0	42	0.42
1	42	0.42
2	52	0.52
3	51	0.51
⋮	⋮	⋮
101	1	0.001
102	2	0.002
⋮	⋮	⋮
181	1	0.001
188	1	0.001

Tabella 3.1: Distribuzione del grado dei nodi: k grado del nodo (numero di amici), *ripetizioni* numero di nodi con grado k, $p(k) = ripetizioni/N$ probabilità che un nodo del grafo G_N abbia esattamente grado k.

L'andamento lineare nella scala logaritmica implica che la distribuzione di probabilità $P(k)$ segua la legge della potenza che abbiamo visto nell'equazione 2.10. Il *fitting* dei dati ci restituisce $\gamma_N = 1,358$ che è più basso rispetto al valore del modello *scale-free* di Barabási e Albert ($\gamma_{BA} = 3$) e questo significa che nel nostro sottografo G_N la probabilità di trovare *hub* è molto più alta. Nel nostro caso abbiamo visto come esistano nodi collegati a quasi il venti per cento della rete: a meno che due nodi non siano già amici è molto probabile che siano collegati allo stesso *hub*, così la loro distanza sarà solamente 2.

Per calcolare il coefficiente di aggregazione C_N del sottografo G_N si è visto in 1.3.4 che bisogna prima conoscere C_i, coefficiente del nodo i, e poi farne la media. Siccome C_i è dato solo dagli elementi della diagonale di A^3 e dal grado k_i del vertice, non dobbiamo calcolare tutta la Matrice A^3, che per queste dimensioni è una operazione lunga anche per un calcolatore, ma solo ricavare la formula degli elementi di $diag(A^3) = (a_1^3, a_2^3, \ ... \ , a_N^3)$:

1. chiamiamo a_{ij}^2 gli elementi di A^2 (notare che $a_{ii}^2 = k_i$ grado del nodo i),

$$a_{ij}^2 = \sum_{k=1}^{N} a_{ik} a_{kj}, \quad A = (a_{ij})$$

2. gli elementi a^3_{ij} della matrice $A^3 = A^2\ A$ sono dati da

$$a^3_{ij} = \sum_{m=1}^{N} a^2_{im} a_{mj} = \sum_{m=1}^{N} (\sum_{k=1}^{N} a_{ik} a_{km}) a_{mj}$$

3.

$$a^3_i = a^3_{ii} = \sum_{m=1}^{N} (\sum_{k=1}^{N} a_{ik} a_{km}) a_{mi} \in diag(A^3)$$

Andiamo ora a sostituire nella formula $C_i = 2\ E_i/k_i(k_i - 1)$ i valori trovati per ogni nodo i, sapendo che $2\ E_i = a^3_i$. Facendo la media si trova per il grafo G_N un coefficiente di aggregazione $C_N = 0,357$. Possiamo ora confrontare C_N con il coefficiente teorico di un grafo aleatorio della stessa dimensione N e lo stesso grado medio $< k >= (\sum_{i=1}^{N} k_i)/N = 27,417$ del il nostro G_N. Viene che $C_{rand} =< k > /N = 0,0274$, che è circa $1/13$ del coefficiente C_N da noi ottenuto. Il primo ingrediente del *piccolo mondo* c'è: una alta concentrazione di relazioni tra gli amici di ogni persona.

Ci aspettiamo ora che anche il grado di separazione medio $< l >_N$ di G_N sia piccolo, così che anche la seconda caratteristica dei piccoli mondi sia rispettata dalla nostra rete.

Dobbiamo per primo calcolare $l(i, j)\ \forall\ i, j \in V_N$, cioè il numero minimo di archi da percorrere per raggiungere j partendo da i. Questi valori saranno memorizzati in una matrice $D \in \mathbb{M}_{N\times N}$. Siccome G_N non è orientato anche questa matrice sarà simmetrica come A, infatti ogni $i - j$ cammino può essere percorso in entrambe le direzioni, allora $l(i, j) = l(j, i)$.

Se una coppia di vertici non è connessa le verrà assegnato un valore $+\infty$ nella matrice D, ma questo dato verrà saltato al momento del calcolo di $< l >_N$ essendo questa la media di tutti i valori $l(i, j)$ calcolati. Anche i valori $l(i, i) = 0$ non verranno presi in considerazione . Per il calcolo di $l(i, j)$ si possono usare diversi algorimti, quello usato da noi è il Flooyd - Warshall (Appendice A), adatto a trattare le matrici di adiacenza.
Possiamo anche vedere la *distribuzione dei gradi di separazione* (Tabella 3.2 e Figura 3.2) contando quante coppie hanno la stessa distanza l, chiamiamo $r(l)$ questo numero. Si contano però solo i valori di D sopra o sotto la diagonale. Per avere una misura di probabilità dovremo poi dividere questi risultati per L_N dove L_N è dato da

$$L_N = \sum_{t=1}^{max(l)} r(t)$$

l	$ripetizioni = r(l)$	$p(l) = ripetizioni/L_N$
1	13708	0,030092
2	129184	0,283586
3	188249	0,413245
4	102729	0,225511
5	17752	0,038969
6	3072	0,006744
7	743	0,001631
8	94	0,000206
9	6	0,000013
10	1	0,000002

Tabella 3.2: Distribuzione del grado di separazione: l distanza tra due nodi; *ripetizioni* $= r(l)$ numero di coppie a distanza l; $p(l) =$ *ripetizioni*$/L_N$ probabilità che una coppia di nodi di G_N siano a distanza l, $L_N = \sum_{t=1}^{max(l)} r(t)$

Si vede in Figura 3.2 come questi dati rispecchiano fedelmente una distribuzione Gaussiana del tipo

$$P(x) = a\ e^{\left(\frac{x-b}{c}\right)^2}$$

Il coefficiente $b = 2,911$ della *Gaussiana* rappresenta i valore medio di $P(l)$ ed è molto vicino alla valore medio reale $< l >_N = 2,988$ calcolato da $l(i,j)$. Possiamo quindi affermare che in media due persone del network G_N dovranno fare solo tre passaggi per mettersi in contatto attraverso la rete delle amicizie, ma soprattutto il 95% delle coppie ha al massimo 4 gradi di separazione e il 99% ne ha al massimo 5.

Abbiamo visto quindi che nel grafo G_N coesistono un alto coefficiente di aggregazione e il grado di separazione medio è molto basso. G_N ha quindi tutte le caratteristiche di un piccolo mondo e in più la distribuzione dei gradi dei vertici segue la legge della Potenza.

Siamo quindi in presenza di una rete estremamente efficiente ed efficace per veicolare messaggi e informazioni a più persone possibili.

3.2.2 Crescita dei sottografi di G

Vediamo ora come questa rete si sviluppa ripetendo generazioni casuali dei sottografi di G. All'inizio N, dimensione del grafo, sarà 100, poi 200, e così via fino a 1000. Per ogni dimensione vengono generati casualmente 100 sottografi

$$G_N^{\,i}\ i \in \{1, 2, 3..., 100\}\ e\ N \in \{100, 200, 300, ..., 1000\}\ .$$

Per ognuno di questi G_N^i calcoliamo, come abbiamo visto nel capitolo precedente, i valori del grado di separazione medio, l'esponente γ della legge della potenza della distribuzione dei gradi, il coefficiente di aggregazione e il numero medio di amici $< k >$ dei nodi.

A questo punto, facendo variare i per N fissato, troviamo il valore dei nostri parametri in funzione di N. Ad esempio possiamo calcolare così il coefficiente di aggregazione C_N

$$C_N = \frac{1}{100} \sum_{i=1}^{100} C_N^i$$

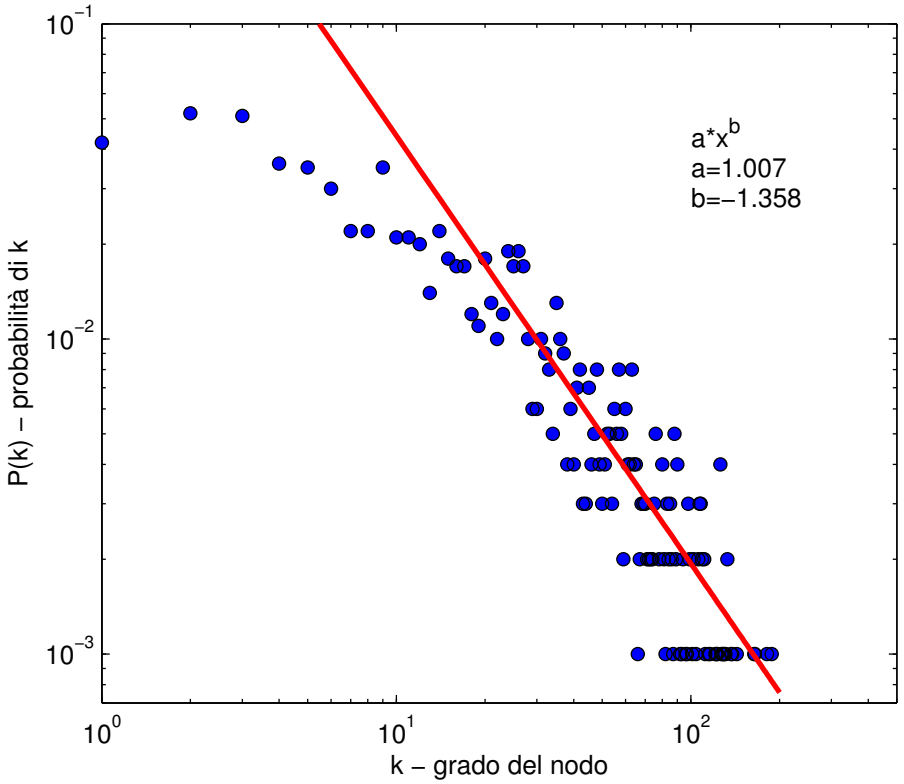
I dati così ricavati sono nella Tabella 3.3

N	γ_N	$< l >_N$	C_N	$< K >_N$
100	1,3042	3,68443	0,18041	2,51210
200	1,3643	3,76875	0,24930	5,11565
300	1,3867	3,55726	0,28275	7,49325
400	1,3473	3,35261	0,30448	10,17958
500	1,3150	3,25640	0,32142	12,76934
600	1,3092	3,18947	0,33041	14,93637
700	1,2906	3,10617	0,34165	17,59526
800	1,2694	3,06714	0,34883	20,16402
900	1,2626	3,02974	0,35231	22,51245
1000	1,2441	3,00077	0,35910	25,22215

Tabella 3.3: N: numero di nodi - γ_N: esponente della Legge della potenza $P(k) \approx k^{-\gamma}$ - $< l >_N$: grado di separazione medio - C_N: coefficiente di aggregazione - $< k >_N$: numero medio di relazioni (amici)

Abbiamo così trovato i dati relativi allo sviluppo dell'*hub* di partenza, inizialmente per un numero piccolo di persone coinvolte, poi aumentandolo fino ad arrivare a decuplicarlo. Questi sottografi di 1000 profili, rispetto al numero di quelli presenti nell'intero social network *Facebook*, ne rappresentano una micro porzione, tuttavia l'andamento dei dati si è rivelato molto buono tanto da rendere plausibile l'ipotesi che il funzionamento sia molto simile per porzioni più ampie di network. Nel prossimo capitolo vedremo i risultati dell'inferenza statistica dei nostri dati.

a)



b)

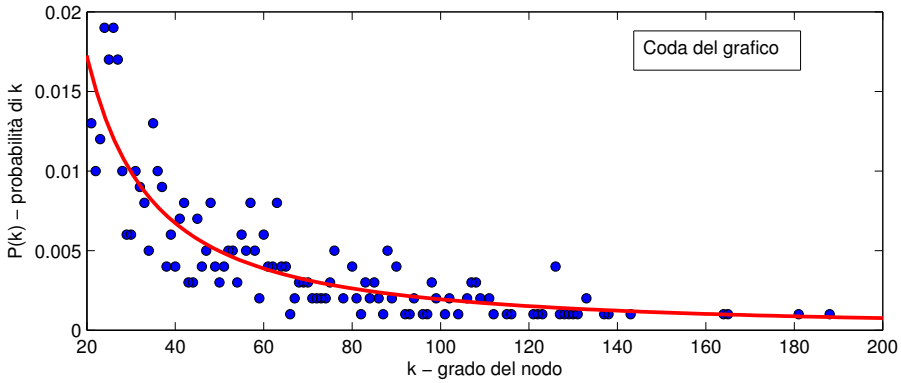


Figura 3.1: a) Grafico in scala logaritmica dei dati della Tabella 3.1 che si riferiscono alla distribuzione dei gradi dei nodi di G_N e in rosso la funzione che approssima i valori che, essendo una retta in questo grafico, è una potenza $k^{-\gamma}$ b) particolare della *coda* del grafico, si vede come il Fitting dei dati relativi alla Tabella 3.1 è ben approssimata dalla legge della Potenza

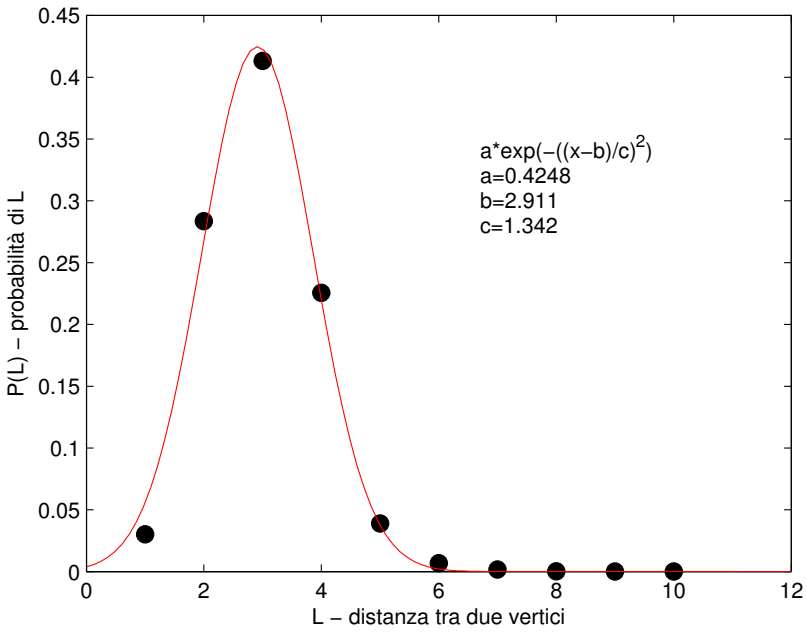


Figura 3.2: Fitting Gaussiano dei dati relativi alla Tabella 3.2

Capitolo 4

Conclusioni

Quelli che presentiamo ora sono i grafici dei dati della Tabella 3.3 dove è stato fatto anche un *fitting* dei dati con il programma di calcolo *MATLAB*.

4.1 Il coefficiente di aggregazione

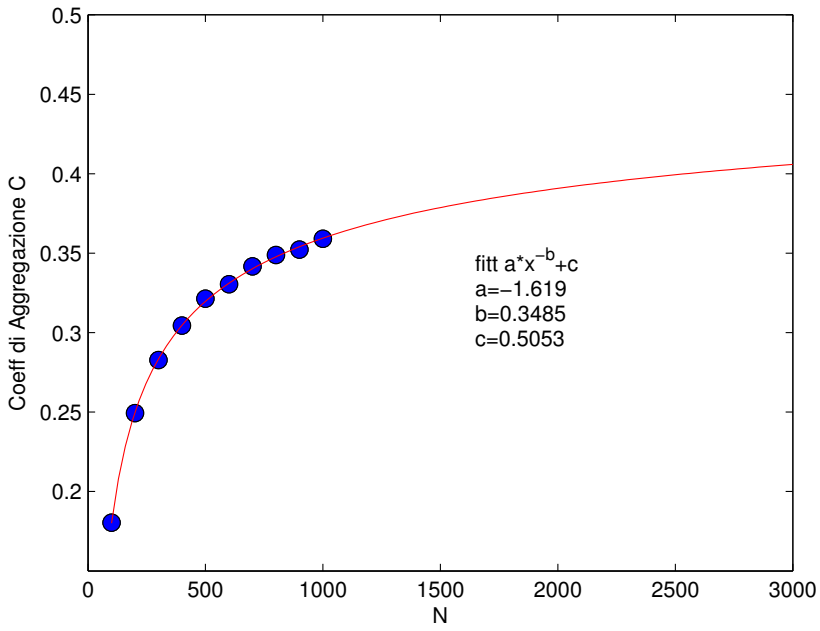


Figura 4.1: Fitting di C_N relativo alla Tabella 3.3. A crescere di N, C_N si stabilizza e si avvicina asintoticamente al valore $c = 0,5053$.

Vediamo in Figura 4.1 che C_N cresce con N, ma sempre più lentamente, e asintoticamente si avvicina al valore 0,5053. Sembra quindi che quando il numero di persone aumenta, la rete di amicizie diventi sempre più fitta. Questo può essere dovuto al fatto che *Facebook* suggerisce ai nuovi membri della comunità le prime amicizie, incrociando i dati sulla provenienza, sugli studi e sugli interessi personali, ma anche sulle amicizie comuni dei profili già nostri amici.

4.2 Grado di separazione Medio

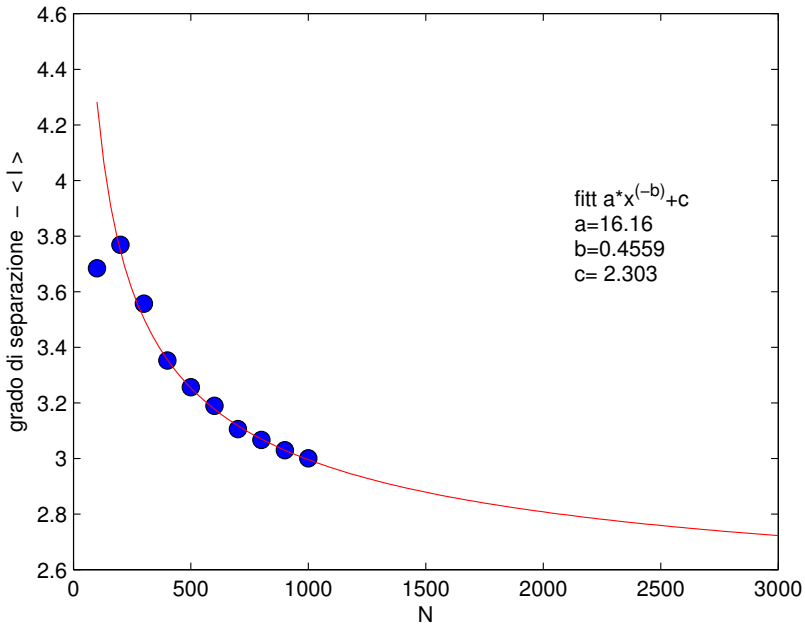


Figura 4.2: Fitting di $< l >_N$ relativo alla Tabella 3.3. In ascissa N e in ordinata il valore $< l >_N$. Escludendo i valori bassi di $< l >_N$, i dati risultano rispecchiare bene la legge della potenza $ax^{-b} + c$, quindi $< l >_N$ tende asintoticamente al valore $c = 2,303$

La Figura 4.2 ci mostra come il grado di separazione medio si abbassa velocemente con l'aumentare di N. Il valore di $< l >_N$ si avvicina a 2,303 seguendo un andamento a potenza, anche se la rete aumenta la sua dimensione, sembra rimanere invariata la distanza tra due nodi qualsiasi. Questo fenomeno in altri network è spiegato dalla presenza dei grandi *hub* che mantengono ben interconnessa la rete e senza i quali le distanze aumenterebbero molto.

4.3 Esponente γ - Distribuzione dei gradi

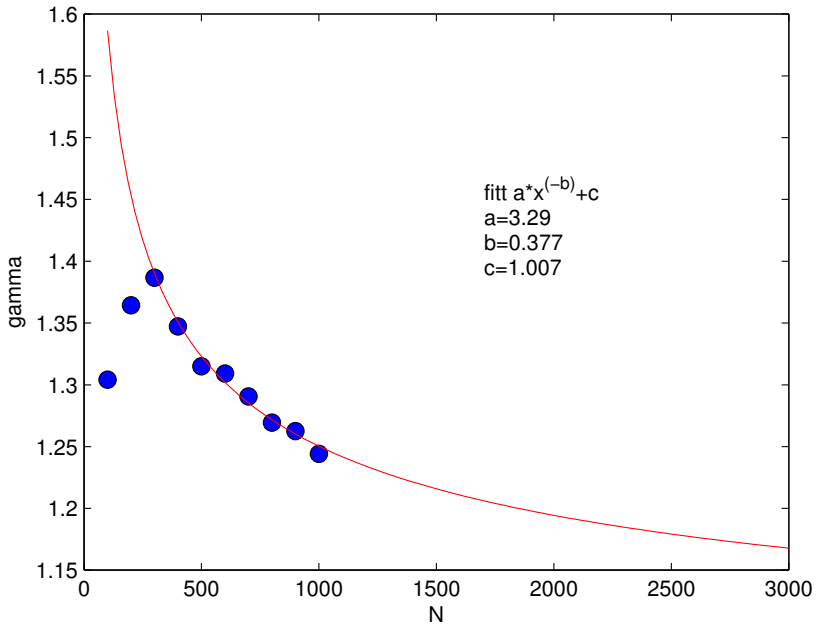


Figura 4.3: Fitting di γ_N relativo alla Tabella 3.3. In ascissa N e in ordinata il valore γ_N. Escludendo i valori bassi di γ_N, i dati risultano rispecchiare abbastanza bene la legge della potenza $ax^{-b} + c$, quindi γ_N tende asintoticamente al valore $c = 1,007$

L'inferenza statistica dei dati relativi all'esponente della legge di distribuzione $P(k) \sim k^{-\gamma}$ (Figura 4.3) risulta meno buona degli altri casi. Il grafico ci indica comunque che quando il numero di nodi aumenta, aumenta anche la probabilità di trovare *hub* di grosse dimensioni (diminuendo γ), e questo è coerente con le considerazioni fatte su coefficiente di aggregazione e grado di separazione.

4.4 Numero medio di amici

Il numero di amici medio nei sottografi di G aumenta linearmente con N (Figura 4.4), ma non possiamo pensare che questo accada anche per N della dimensione dell'intera rete di *Facebook*. Il numero di utenti registrati a fine 2009 era 350 milioni, dal grafico abbiamo desunto che nel nostro *hub* $< k >_N$ è circa il 2,5% di N, se calcoliamo il 2,5% di 350 milioni risulta 8 milioni e 750 mila, un numero decisamente esagerato. *Facebook* ha pubblicato sul suo sito alcune statistiche, tra

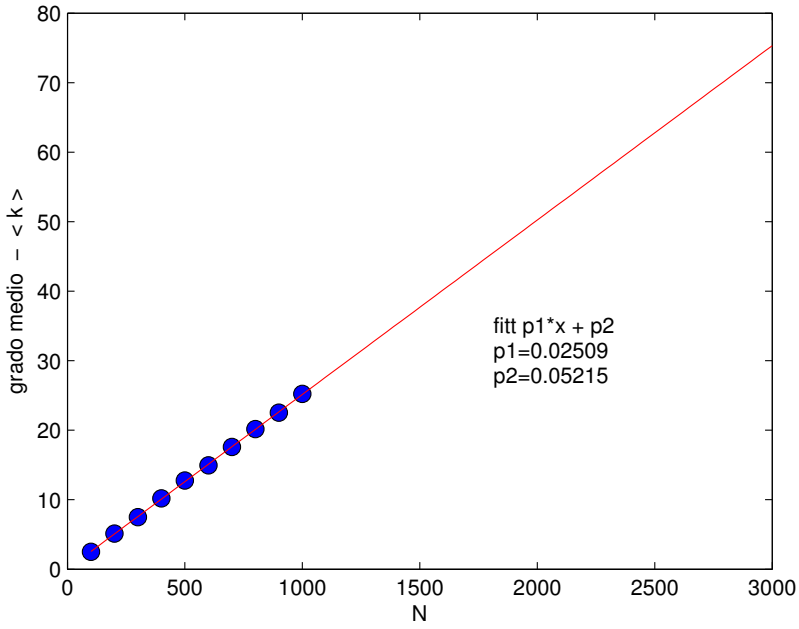


Figura 4.4: Fitting di $< k >_N$ relativo alla Tabella 3.3. Al crescere di N, $< k >_N$ cresce linearmente con coefficiente 0,02509, cioè il numero medio di amicizie corrisponde al 2,5% di N.

cui il numero medio di amici dei suoi profili, che risulta essere 130. Questo è dovuto in parte al fatto che è imposto un limite al numero di amicizie che si possono avere, in parte al fatto che, come dimostrato da studi sociologici[1], difficilmente si possono intrattenere interazioni stabili con più di 150 persone.

[1]Si vedano ad esempio gli studi di Robin Dunbar ed il suo numero *Dunbar's number*

Appendice A

Algoritmo Floyd-Warshall

Il *Floyd-Warshall* è un algoritmo per l'analisi dei grafi che serve a trovare il percorso più breve tra due nodi. Funziona anche in grafi orientati e con archi *pesati*. Una singola esecuzione dell'algoritmo trova tutti i percorsi più brevi tra tutte le coppie di vertici. L'algoritmo fu scoperto da *Robert Floyd* e parallelamente anche da *Stephen Warshall* nello stesso anno, il 1962.

L'idea di base di questo algoritmo è un processo iterativo che, scorrendo tutti nodi, ad ogni passo h costruisce una matrice A_h dove in posizione (i, j) si trova memorizzata la distanza minima (pesata) dal nodo i a quello j attraversando solo nodi di indice minore o uguale ad h, se non vi è collegamento allora nella cella c'è *infinito*. Alla fine (con $h = N =$ numero di nodi) leggendo la matrice si ricava la distanza minima fra vari nodi del grafo.

Premesso ciò, si può dire che il problema si risolve in programmazione dinamica facendo uso teoricamente di una matrice cubica dove le ordinate e le ascisse sono i e j, mentre h è la profondità. Praticamente, invece, possiamo usare una sola matrice reiterandola piano per piano in h. Si comincia col ricavare la matrice di adiacenza A del grafo, dove non c'è collegamento mettiamo *infinito* al posto di 0, avremo così ricavato A_0. Poi, basandoci su di essa, in ogni passo successivo h si esaminano le celle (i, j) della matrice A_{h-1}: se la distanze tra il nodo i e il nodo in esame h sommata alla distanza tra h e j è minore della distanza tra i e j trovata fino al passo $h-1$, allora si sostituisce quest'ultimo con la precedente somma nella nuova matrice A_h.

Formalmente: se

$$A_{h-1}[i, h] + A_{h-1}[h, j] < A_{h-1}[i, j]$$

allora

$$A_h[i, j] = A_{h-1}[i, h] + A_{h-1}[h, j] \ ;$$

altrimenti

$$A_h[i, j] = A_{h-1}[i, j] \ .$$

Con una piccola modifica dell'algoritmo, è anche possibile ricostruire il percorso del cammino più breve, cosa che la formulazione originale non prevede. Non è nemmeno necessario memorizzare l'intero percorso per ogni coppia di nodi, che sarebbe molto costoso in termini di memoria. Per ogni nodo è necessario memorizzare solamente il primo nodo del percorso che porterebbe ad un qualsiasi nodo desiderato. Allora queste informazioni sono tutte raccolte in una matrice $N \times N$ chiamata *next*, dove $next[i][j]$ rappresenta il primo nodo da raggiungere per arrivare in j partendo da i.

Questo schema si implementa facilmente quando la matrice A_h viene aggiornata. Alla fine entrambe le matrici A_N e *next* saranno complete, ad ogni *infinito* corrisponderà un puntatore vuoto nella matrice *next*. Il percorso da i a j, per come è stata definita *next*, avrà come primo vertice $next[i][j]$, poi il procedimento verrà iterato cercando il percorso tra $next[i][j]$ e j, fino ad arrivare al nodo j.

Pseudo codice del programma

```
 1 procedure FloydWarshallWithPathReconstruction ()
 2    for k := 1 to n
 3       for i := 1 to n
 4          for j := 1 to n
 5             if path[i][k] + path[k][j] < path[i][j] then
 6                path[i][j] := path[i][k]+path[k][j];
 7                next[i][j] := k;
 8
 9 procedure GetPath (i,j)
10    if path[i][j] equals infinity then
11      return "no path";
12    int intermediate := next[i][j];
13    if intermediate equals 'null' then
14      return " ";   /* there is an edge from i to j, with no vertices
                                between, only if the graph is direct */
15   else
16      return GetPath(i,intermediate) + intermediate + GetPath(intermediate,j);
```

Appendice B

Listato del Programma

Ecco il listato del programma in linguaggio C. Viene lanciato da terminale (sistemi UNIX) e come argomento necessita tre file di testo: il primo dove è specificato l'elenco degli amici, il secondo dove in ogni riga vi sono due nomi di amici ed è il file delle relazioni, infine il terzo file è l'output del programma dove viene salvata la tabella dei dati relativi ad ogni sottografo G_N^i generato casualmente.

```
//Versione 4.0

#include <stddef.h>
#include <stdio.h>
#include <time.h>
#include <string.h>
#include <math.h>
#include <time.h>

#define dim 1000
#define passo 100
#define ripetizioni 100
#define HUGE_VAL 1430
//2322
//1430

FILE *in, *out, *rel; //in:=lista amici rel:=lista relazioni out:=Output

/*
typedef struct {
char pers1[50];
char pers2[50];
} relazione;
```

```
*/
typedef struct {
char nome[50]; //50B
int num_amici; //2B
float claust;
float dist_media; //4B
int indice; //2B
} persona;
typedef struct {
double a;
double b; //esponete della potenza
double sigma_a;
double sigma_b;
int sum1;
int n;
} regr;




void swap(persona *pmax, persona *p)
{
persona m;
m=*p;
*p=*pmax;
*pmax=m;
}


void genera_amici(int i_main,persona nodi[],persona amici[])
{
int i,j,r;
int list[2*dim];

for(i=0;i<2*dim;i++) list[i]=i;

for(i=0;i<i_main;i++) {
r=rand()%(2*dim-i);
amici[i]=nodi[list[r]];
amici[i].indice=i;
//printf("%d\n",r);

for(j=r;j<2*dim-1;j++)
```

```
list[j]=list[j+1];
}
printf("\n\n%d\n\n",i_main);

}



void bouble_sort(int n,persona list[])
{
int stop=1, k=n-1, i;
while (stop==1 && k>0)
{
stop=0;
for (i=0; i<k; i++)
{
if (strcmp(list[i].nome , list[i+1].nome) > 0)
{
swap(&(list[i]), &(list[i+1]));
stop=1;
}
}
k--;
}
}


void conta_amici(int n,persona list[],int mat[][dim])
{
int i=0,j=0;
for(i=0;i<n;i++)
{
list[i].num_amici=0;

for(j=0;j<n;j++) {

list[i].num_amici=list[i].num_amici+mat[list[i].indice][list[j].indice];

}
}
}
```

```
void coeff_aggregazione(int n, persona list[],int mat[][dim])
{
int i,j,k;

for(i=0;i<n;i++)
{
list[i].claust=0.0;

for(j=0;j<n;j++)
for(k=0;k<n;k++)
list[i].claust+=(float)(mat[list[i].indice][list[j].indice]*
mat[list[j].indice][list[k].indice]*
mat[list[k].indice][list[i].indice]);


if(list[i].claust!=0 && list[i].num_amici>1)
list[i].claust/=(float)(list[i].num_amici*(list[i].num_amici-1));
}

}




void sort_amici(int n,persona list[])
{
int stop=1, k=n-1, i;
while (stop==1 && k>0)
{
stop=0;
for (i=0; i<k; i++)
{
if (list[i].num_amici<list[i+1].num_amici)
{
swap(&(list[i]), &(list[i+1]));
stop=1;
}
}
k--;
}
}
```

```
float media_amici(int n,persona list[])
{
int i=0;
float media=0;
for(i=0;i<n;i++)
media=media+list[i].num_amici;
media=media/n;
return(media);
}

float dev_std_amici(int n,persona list[])
{
int i=0;
float media=0,dev=0;

for(i=0;i<n;i++)
{
media=media+list[i].num_amici;
dev=dev+pow((list[i].num_amici),2);
}

//dev=sqrt(dev/n-pow((media/n),2));
dev=(sqrt(n*dev-pow(media,2)))/(n-1);
//printf("%f\n",media/n);

return(dev);
}

float media_claust(int n,persona list[])
{
int i=0;
float media=0.0;
for(i=0;i<n;i++)
media=media+list[i].claust;
media=media/n;
return(media);
}

float dev_std_claust(int n,persona list[])
{
int i=0;
float media=0,dev=0;
```

```
for(i=0;i<n;i++)
{
media=media+list[i].claust;
dev=dev+pow((list[i].claust),2);
}

//dev=sqrt(dev/n-pow((media/n),2));
dev=(sqrt(n*dev-pow(media,2)))/(n-1);
//printf("%f\n",media/n);

return(dev);
}



//2.0 restituisce la distanza media di una una persona
//dalle altre e la dist media dei nodi del grafo
float distanza_media(int n,persona list[],int dist_mat[][dim])
{
int k,i,j;
float l=0.0;
int l_count=0;

for(i=0;i<n;i++)
{
if(list[i].num_amici>0)
{
list[i].dist_media=0;
k=0;
//-1 perchè il ciclo non salta il valore della distanza 0 sulla diagonale
for(j=0;j<n;j++)
{
if (dist_mat[list[i].indice][j]<HUGE_VAL && dist_mat[list[i].indice][j]>0)
{
k++;
list[i].dist_media=list[i].dist_media+dist_mat[list[i].indice][j];
l+=dist_mat[list[i].indice][j];
l_count++;
}
}
list[i].dist_media=list[i].dist_media/k;
}
else list[i].dist_media=HUGE_VAL;
```

```
}
return(l/(float)l_count);
}



void vett_distanze(int n,int dist_mat[][dim],int vett[])
//in vett aggiungo 1 alla posizione corrispondente alla distanza
{ //in posizione 0 il contatore per la normalizzazione
int i,j;

// printf("\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n ");

for(i=0;i<n;i++) vett[i]=0; //inizializzo il vettore

for(i=0;i<n;i++)
for(j=i;j<n;j++)
{
if (dist_mat[i][j]<HUGE_VAL && dist_mat[i][j]>0)
{
// printf("%d ",vett[dist_mat[i][j]]);
vett[dist_mat[i][j]]=vett[dist_mat[i][j]]+1;
vett[0]++;
// printf("%d ",vett[dist_mat[i][j]]);
}
}
}


void vett_amici(int n,persona list[],int vett[])
{
int i;

for(i=0;i<n;i++) vett[i]=0;

for(i=0;i<n;i++)
vett[list[i].num_amici]++;

}


/* * Descrizione : calcola il fit lineare
 * * y=a+bx di un set di dati (x_i,y_i) in scala Logaritmica
```

```
 * * e riporta la funf ad una potenza y=e^a*x^b
 */

void lin_fit(int n, int vett[], regr *dati_out)
{
int i;

int sum_1 ; /* # coppie dati */
double x, y; /* coppie di dati */
double sum_x ,sum_y ; /* variabili cumulate */
double sum_xx ,sum_xy , sum_yy ;  /* variabili cumulate */
double coef_a ,coef_b , det ; /* coefficienti retta */
double sigma_a ,sigma_b ; /* errori */
double  chi_2 ;


/* inizializzazione variabili cumulate */
sum_1 = 0; /*sum1*/
sum_x = 0.0; /*sumx*/
sum_y = 0.0; /*sumy*/
sum_xx = 0.0; /*sumxx*/
sum_xy = 0.0; /*sumxy*/
sum_yy = 0.0; /* sum yy */

for(i=5;i<n;i++)
if(vett[i]!=0) {
x=log((double)i);
y=log((double)vett[i]);
++sum_1;
sum_x += x;
sum_y += y;
sum_xx += x * x;
sum_xy += x * y;
sum_yy += y * y;

// printf("\nsum_x=%g \t x=%g \t y=%g\n",sum_x,x,y);
}

// printf("\nSum_1=%d\n\n",sum_1);

if (sum_1 == 1) {
printf("\nNon abbastanza dati\n\n");
exit (1);
```

```
}


/* Coefficienti retta a e b */
det = sum_xx*sum_1 - sum_x*sum_x;
coef_a = (sum_xx*sum_y - sum_x*sum_xy)/det;
coef_b = (sum_xy*sum_1 - sum_x*sum_y)/det;

printf("\ndet=%.3g\n\n",det);

printf("\ncoef_a=%.3g\n\n",coef_a);

printf("\ncoef_b=%.3g\n\n",coef_b);

//det = (sum_xx * sum_1) ? (sum_x)*(sum_x) ;//
//coef_a = (sum_xx * sum_y ? sum_x * sum_xy ) / det;
//coef_b = (sum_xy * sum_1 ? sum_x * sum_y ) / det;

/* Errori sulla stima */
if (sum_1 == 2) {
sigma_a = 0.0;
sigma_b = 0.0;
}
else {
chi_2 = sum_yy
+ coef_a * coef_a * sum_1
+ coef_b * coef_b * sum_xx ;

chi_2 -= 2.0*(coef_a*sum_y + coef_b*sum_xy - coef_a*coef_b*sum_x);
chi_2 /=(double)(sum_1-2);

sigma_a = ( sum_xx / det ) * chi_2;
sigma_b = ( (double) sum_1 / det ) * chi_2;
}


//salvo i dati nell'Output

dati_out->a=exp(coef_a);
dati_out->b=coef_b;
dati_out->sigma_a=sigma_a;
dati_out->sigma_b=sigma_b;
dati_out->sum1=sum_1;
```

```
dati_out->n=n;

}




// -_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-
// ------------- Inizio programma ---------------
// -_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-_-

int main (int argc, const char * argv[]) {

int i=0,n, j,k, count=0;    //cout:=numero di relazioni vagliate
int stime,ltime;
char pers1[50],pers2[50];
int i_main,j_main; //contatori sovraprogramma
int edge;

persona nodi[dim*2];
persona amici[dim];

int mat[dim][dim]={0};
//matrice delle adiacenza
int dist[dim][dim];
//matrice delle distanze
int pred[dim][dim];
//matrice per la ricostruzione del percorso più breve tra 2 elementi
int cont_dist[dim]={0};
//vettore che conta le ripetizioni delle distanze
int cont_amici[dim]={0};
//vettore che conta le ripetizioni del numero di amici

float l; //media delle distanze tra due nodi, diametro del grafo


//regr funz[ripetizioni*dim/passo];
regr funz;

//relazione connection[65000],p;


if ((in=fopen(argv[1],"r"))==NULL)
```

```
{
printf("errore nel file di input\n");
exit(1);
}
//if ((rel=fopen(argv[2],"r"))==NULL)
// {
// printf("errore nel file di relazioni\n");
// exit(1);
// }
if ((out=fopen(argv[3],"w"))==NULL)
{
printf("errore nel file di output\n");
exit(1);
}

//inizializzo variabili random
ltime=time(NULL);
stime=ltime/2;
srand(stime);


while (fscanf(in,"%s",&nodi[i].nome)>0 && i<dim*2)
{
i++;
}

n=i;
i=0;
j=0;

//RIPETIZIONE DEL PROGRAMMA

for(i_main=100;i_main<=dim;i_main+=passo)
{
for(j_main=0;j_main<ripetizioni;j_main++)
{

genera_amici(i_main,nodi,amici);
//genera un sottoinsieme casuale di nodi con cardinalità i_main

n=i_main; //inizializzo variabili controllo
count=0;
edge=0; //numero di relazioni
```

```
for(i=0;i<n;i++) //inizializzo la matrice delle addiacienze
for(j=0;j<n;j++)
mat[i][j]=0;

//

//rendo attivo il file delle relazioni

if ((rel=fopen(argv[2],"r"))==NULL){
printf("errore nel file di relazioni\n");
exit(1);
}

while (fscanf(rel,"%s %s",&pers1,&pers2)>0)
//genero la matrice delle addiacienze
{
i=0;
j=0;

while (strcmp(pers1,amici[i].nome)!=0 && i<n)
i++;

while (strcmp(pers2,amici[j].nome)!=0 && j<n)
j++;

if (i<n && j<n)
{
mat[i][j]=1;
mat[j][i]=1;
edge++;
}
count++;
}



//bouble_sort(n,amici);
//printf("mat completa n=%d\n",n);

conta_amici(n,amici,mat);

coeff_aggregazione(n, amici, mat);
```

```
sort_amici(n,amici);


//algorithm initialization

  for (i=0; i < n; i++) {
    for (j=0; j < n; j++) {
      if (mat[i][j] != 0)
dist[i][j] = mat[i][j];
      else
dist[i][j] = HUGE_VAL; //disconnected

      if (i==j)  //diagonal case
dist[i][j] = 0;

      if ((dist[i][j] > 0) && (dist[i][j] < HUGE_VAL))
pred[i][j] = i;
    }
  }


  //Main loop of the algorithm
  for (k=0; k < n; k++) {
    for (i=0; i < n; i++) {
      for (j=0; j < n; j++) {
if (dist[i][j] > (dist[i][k] + dist[k][j])) {
  dist[i][j] = dist[i][k] + dist[k][j];
  pred[i][j] = k;
//   printf("updated entry %d,%d with %d\n", i,j, dist[i][j]);
}
      }
    }
  }




l=distanza_media(n,amici,dist);
//per ogni persona calcola la distaza media dai raggiungibili

vett_distanze(n,dist,cont_dist);
//conta le ripetizioni dei valori delle distanze nella matrice dist
```

```
vett_amici(n,amici,cont_amici);
//conta le ripetizioni dei valori del numero delle amicizie

lin_fit(n,cont_amici,&funz);
//Regressione logaritmica



//-------------------------------
//--- STAMPA SU FILE DEI DATI ---
//-------------------------------

if(i_main==100 && j_main==0)
fprintf(out,"n gamma diam C sigmaC <K> sigma<K> edge max(K_i)\n");

//stampa info tabella
//Intestazione tabella |n|valore esponente|diametro|
//coefficiente di aggregazione|dev standard sul coef aggr|
//media num amicizie|dev standard su num media amicizie|
//num di relazioni|max amicizie|

fprintf(out,"%d %.3g %.3f %.3f %.3f %.3f %.3f %d %d\n",
          n,funz.b,l,media_claust(n, amici),dev_std_claust(n,amici),
          media_amici(n,amici),dev_std_amici(n,amici),
          edge,amici[0].num_amici);


fclose(rel);
}
}


fclose(in);
fclose(out);


return 0;
}
```

Bibliografia

[1] Mark Buchanan *Nexus, perchè la natura, la società, l'economia, la comunicazione funzionano allo stesso modo*, Milano, Mondadori, 2003, Cap II.

[2] Duncan J. Watts e Steven H. Strogatz, *Collective dynamics of small-world networks*, Nature, 393, 1998, pp 440-42.

[3] Reka Albert e Albert-Laszló Barabási *Statistical machanics of complex networks*, Reviews of Modern Phisics Volume 74, January 2002, pp 47-94.

[4] Barrat e Weigt, 2000, Eur, Phys, J. B 13, 547.

[5] Bollobàs B., 1981, Discrete Math. 33, 1.

[6] Albert-László Barabási e Reka Albert, *Emergence of Scaling in Random Networks*, 1999, Science 286, 509.

Ringraziamenti

Ringrazio il Prosessor Pierluigi Contucci, che mi ha ben guidato nella ricerca dei materiali e nello sviluppo di questa tesi.

Ringrazio Cecilia Vernia, ricercatrice presso l'Università degli studi di Modena e Reggio Emilia, che mi ha efficacemente introdotto ai pacchetti di MATLAB per il fitting dei dati.

Ringrazio il Circolo Arci Accatà, che ha messo a disposizione il suo profilo *Facebook* per l'analisi dei dati.

Ringrazio Jimmy Wales e Larry Sanger per la loro geniale creazione, che regala ottimi spunti per iniziare una buona ricerca.

Ringrazio tutta la mia famiglia, quella non tradizionale.