UNIVERSITÀ DEGLI STUDI DI TORINO
Facoltà di Scienze Matematiche Fisiche e Naturali
Corso di Laurea Magistrale in Informatica

Riassunto di

Reti Complesse

Enrico Mensa

Basato sulle lezioni di: *Ruffo Giancarlo, Sereno Matteo*

Indice

1 Introduzione **1**

1.1 Perché le reti complesse? 1

1.1.1 La scienza della complessità 2

1.1.2 Analisi di reti complesse 2

1.1.3 Le reti sociali 3

1.1.4 Le reti di informazioni 4

1.2 Richiami della teoria dei grafi 6

1.2.1 Tipi di rete 7

1.2.2 Rappresentazione di una rete 7

1.2.3 Caratterisiche di una rete 8

2 Strumenti di analisi delle reti **13**

2.1 Un tour fra i tool 13

2.2 Ghepi 14

2.3 R e igraph 14

2.3.1 Installazione di igraph 14

2.3.2 Stampare e visualizzare grafi 15

2.3.3 Estrazione di singole proprietà 15

2.3.4 Grado dei nodi del grafo 16

2.3.5 Creare altre strutture 16

2.3.6 Creare da matrice di adiacenza 17

2.3.7 Attributi 17

2.3.8 Visualizzazione con plot e (tkplot) 18

2.3.9 Layout dei plot 18

2.3.10 Reti egocentriche 19

2.3.11 Altre funzionalità di R 19

2.3.12 Istruzioni base 20

2.3.13 Operatori aritmetici per vettori 20

2.3.14 Media e varianza 21

2.3.15 Espressioni regolari 21

2.3.16 Vettori di stringhe 21

2.3.17 Conversioni 22

2.3.18 Operazioni su matrici 23

3 Misurare la struttura di una rete **25**
3.1 Misure di centralità fondamentali . . . 25
3.1.1 Degree centrality . . . 25
3.1.2 Closeness centrality . . . 26
3.1.3 Betweenness centrality . . . 27
3.2 Misure di centralità più sofisticate . . . 27
3.2.1 Hubs and Authorities (HITS) . . . 28
3.2.2 Page Rank . . . 31
3.3 Gruppi di vertici . . . 32
3.3.1 Clique . . . 32
3.3.2 K-plex . . . 32
3.3.3 K-core . . . 33
3.4 Transitività aka coefficiente di clustering . . . 33
3.4.1 Definizioni di transitività . . . 33
3.4.2 Coefficiente di clustering locale . . . 34
3.5 Reciprocità e segni sugli archi . . . 34
3.5.1 Misurare la reciprocità . . . 34
3.5.2 Archi con segno . . . 34
3.6 Misure di similarità . . . 35
3.6.1 Cosine similarity . . . 35
3.6.2 Coefficienti di Pearson . . . 37
3.6.3 Equivalenze regolari . . . 38
3.7 Homophily e Assortative Mixing . . . 38
3.7.1 Assortative Mixing discreto . . . 39
3.7.2 Assortative Mixing scalare . . . 41
3.7.3 Assortative mixing per grado . . . 42
3.7.4 Influence . . . 43

4 Memorizzare le reti **45**
4.1 Strutture dati . . . 45
4.1.1 Matrici . . . 45
4.1.2 Liste di adiacenza . . . 46
4.1.3 Alberi di adiacenza . . . 46
4.1.4 Heap e ulteriori strutture . . . 47

5 Struttura di grandi reti **49**
5.1 Introduzione . . . 49
5.2 Componenti . . . 49
5.3 Cammini, distanze e diametro . . . 51
5.3.1 Il diametro . . . 51
5.3.2 Effetto funneling . . . 52
5.4 Distribuzione del grado . . . 52
5.4.1 Introduzione: distribuzioni frequenti e asimmetria . . . 53
5.4.2 Leggi di potenza . . . 55

5.4.3 Determinare una power-law empiricamente 57
5.4.4 Ulteriori misure 60
5.4.5 Proprietà delle power-law 61
5.5 Distribuzioni di altre misure 63
5.5.1 Misure di centralità 63
5.5.2 Coefficiente di clustering 63

6 Modelli per reti **65**
6.1 Introduzione 65
6.1.1 Cosa tratteremo 65
6.1.2 Perché i modelli 65
6.2 I grafi random 66
6.2.1 Definizioni preliminari 66
6.2.2 Distribuzione di grado 67
6.2.3 Media degli archi 67
6.2.4 Grado medio 68
6.2.5 Coefficiente di clustering 68
6.2.6 Andamento della giant component 68
6.2.7 Diametro medio della rete 70
6.2.8 Riflessioni sul modello 71
6.3 Modelli a distribuzione generale dei nodi 71
6.3.1 Un modello non statico 71
6.3.2 L'idea del preferential attachment 72
6.3.3 Master equation 72
6.3.4 Cenni al modello Barabàsi-Albert 75
6.3.5 Power-law e limiti di sviluppo 75
6.3.6 Il vertex copying 76
6.3.7 Reti strategiche 76
6.3.8 Study case: diffusione di una patologia 76
6.4 Modello small world 79
6.4.1 Lo studio dei motifs 79
6.4.2 Il modello (Watts-Strogatz) 80
6.4.3 Grafi random per determinare situazioni small world . 82
6.5 La navigabilità di Kleinberg 83
6.5.1 Ottenere la navigabilità 83
6.6 Epidermiology 84
6.6.1 La malattia 84
6.6.2 Le reti di contatti: modelli epidemici 84
6.6.3 Il modello *SI* 86
6.6.4 Il modello *SIR* 88
6.6.5 Il modello *SIS* 91
6.6.6 Il modello *SIRS* 92
6.6.7 Randomness: i modelli stocastici 93

6.6.8 Modelli epidemici sulle reti . . . 93
6.6.9 Proprietà late-time del modello SIR . . . 95
6.7 Game strategies . . . 97
6.7.1 Esempi introduttivi . . . 97
6.7.2 Giochi su reti . . . 99
6.7.3 Network formation games . . . 101

7 Algoritmi sulle reti 103
7.1 Algoritmi principali . . . 103
7.1.1 Grado e distribuzione del grado . . . 103
7.1.2 Distribuzione su ogni nodo . . . 103
7.1.3 Distribuzione cumulativa . . . 103
7.1.4 Coefficiente di clustering . . . 104
7.1.5 Calcolo dei cammini minimi . . . 106
7.2 Algoritmo di Ford-Fulkerson . . . 106
7.2.1 Idea dell'algoritmo . . . 107
7.2.2 Il grafo residuo . . . 108
7.2.3 Teorema del maxflow-mincut . . . 111
7.2.4 Costo dell'algoritmo . . . 111
7.2.5 Estensioni dell'algoritmo . . . 112

8 Community detection 115
8.1 Introduzione . . . 115
8.1.1 Graph partitioning . . . 116
8.1.2 Community detection . . . 117
8.1.3 Differenze fra partitioning e community detection . . . 118
8.2 Graph partitioning . . . 118
8.2.1 Bisezione: algoritmo di Kernighan-Li . . . 119
8.3 Community Detection . . . 121
8.3.1 Variante di Kernighan-Lin . . . 122
8.3.2 Massimizzazione spettrale della modularità . . . 122
8.3.3 Algoritmo di Girvan-Newman . . . 125
8.3.4 Clustering gerarchico . . . 127
8.3.5 Approcci greedy . . . 128
8.3.6 InfoMap . . . 131
8.3.7 Resoconto sui metodi . . . 132
8.4 Overlapping communities . . . 132
8.4.1 CFinder . . . 133
8.4.2 Altri metodi . . . 134
8.5 Comunità in evoluzione . . . 134

9 Il grafo Laplaciano **137**
9.1 La matrice Laplaciana 137
9.2 Il processo di diffusione e il grafo laplaciano 138
9.3 Gli autovettori del grafo Laplaciano 140
9.4 Calcolo delle componenti e algebraic connectivity 142

Bibliografia **145**

Capitolo 1

Introduzione

1.1 Perché le reti complesse?

Che cosa mette in comune il cinema, la politica, un consiglio aziendale, il terrorismo, un'epidemia, un social web? Sono tutti *gruppi di elementi correlati fra loro*. Tali elementi possiamo facilmente pensarli come nodi di un grafo, collegati grazie ad archi. Stiamo parlando di **reti complesse**.
Consideriamo la rete cinema: avremo che un attore sarà collegato ad un altro attore dall'arco hanno lavorato insieme. Proprio su questo esempio si basa il portale *oracle of bacon*[1] che, dati due attori, individua il cammino minimo per collegarli.
Se proviamo a mettere in connessione Totò e Sam Worthington (due attori di nazionalità diversa e cronologicamente distanti) ci aspettiamo di trovare un path molto lungo data la grande eterogeneità dei due personaggi. Ma invece, il risultato sorprendente è che in soli tre passi abbiamo un collegamento fra i due attori.
Questo fenomeno è detto *small world*: la molteplicità di connessioni fra i nodi rende i cammini molto piccoli.

Scopo della *network analisys* è quello di studiare fenomeni come quello che abbiamo appena osservato. Il pattern small word non è l'unico interessante, possiamo infatti osservare:

- Invarianza di scala (self similarity): un fenomeno osservabile in una scala della rete è osservabile anche in una scala differente. Tale fenomeno è molto presente in natura ed è rappresentabile dai frattale.

- Coefficente di clustering: misura l'importanza di un nodo.

- Mixing patterns

[1] `http://oracleofbacon.org/`

- Misure di centralità: come possiamo definire un nodo importante? Page ranking, conteggio degli archi, capire se un nodo è bridge per altri nodi più importanti (come nell'esempio del cinema).

- Fattori di influenza: grazie ai nodi più importanti si definiscono i fattori di influenza.

1.1.1 La scienza della complessità

La scienza della complessità si occupa di studiare il legame fra elementi piccoli ed il comportamento collettivo di tali elementi. Pensiamo ad esempio al fenomeno delle lucciole: da sole lampeggiano a frequenze differenti ma se le mettiamo nella stessa stanza (c'è contatto visivo) esse inizieranno a lampeggiare tutte all'unisono.

Riduzionismo e Collettivismo. Il *riduzionismo* ci dice come i fenomeni di alto livello possano essere ridotti a fenomeni di livello più basso fino ad arrivare alla meccanica quantistica: siamo molto familiari a questo approccio (basti pensare allo stack TCP/IP).
Ci si è chiesti se fosse possibile il programma inverso, il *collettivismo*. Se così fosse potremmo spiegare ogni fenomeno a partire dalla fisica elementare. Secondo P. W. Anderson questo approccio non è possibile, ovvero, non possiamo prevedere dati gli elementi attuali che cosa saremo in grado di fare nel futuro. Lo stesso inventore di ARPANet ha dichiarato che il suo intento era unicamente quello di trasmettere un flusso di bit da un calcolatore all'altro, di certo non si sarebbe mai immaginato l'enorme applicazione futura che sarebbe scaturita dal suo lavoro.

1.1.2 Analisi di reti complesse

Come possiamo analizzare le reti complesse? Abbiamo innanzi tutto bisogno di *dati*. C'è enorme differenza fra le tecniche usate nel passato e quelle odierne per effettuare *data mining* (collezionare i dati). Nel passato i dati erano raccolti manualmente tramite questionari. Le informazioni raccolte erano spesso rumorose e la definizione dei questionari era molto complessa. Attualmente grazie alla tecnologia è molto più semplice raccogliere dati e questo può, talvolta, essere addirritura un problema (lavorare con grandi quantitò di dati porta problemi computazionali).

Visualizzazione di dati. I dati possono poi essere *visualizzati* in modalità differenti per fornirci i primi indizi da cui iniziare l'analisi. Nella Figura 1.1 vediamo un chiaro effetto di polarizzazione (si tratta di twit relativi a Obama e Romney durante il periodo delle elezioni negli USA).

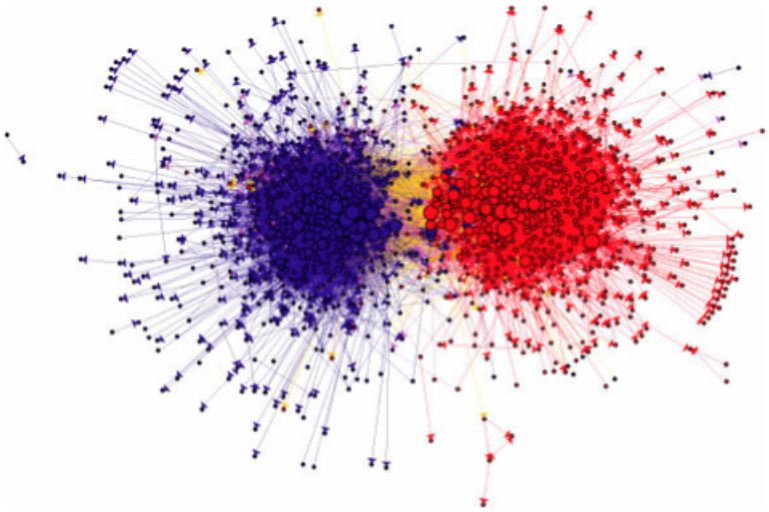


Figura 1.1: Esempio di rappresentazione visuale di una rete (Twitter).

Applicazioni e servizi. Quali applicazioni o servizi possono usufruire del lavoro sulle reti complesse?
Possiamo pensare alla predizione delle epidemie, sistemi di reccomandation, la visualizzazione stessa delle reti, visualizzare dati collegati, etc.

1.1.3 Le reti sociali

Vediamo ora due importanti esempi di esperimenti che hanno aiutato a capire le reti sociali.

Milgram e l'esperimento small word. Stanley Milgram è stato un importante sociologo americano che ha compiuto studi a riguardo dell'autorità e dell'obbedienza fra individui di una società.
A lui si deve la prima definizione di small word, delineata in seguito ad uno dei suoi esperimenti. Tale esperimento consisteva nell'inviare in maniera randomica una lettera in diverse zone degli Stati Uniti. La lettera al suo interno chiedeva di essere inoltrata ad un altro desinatario: nel caso in cui il destinatario citato nella lettera non fosse conosciuto, l'attuale destinatario avrebbe dovuto inoltrare la lettera ad una sua conoscenza che riteneva fosse probabile essere in contatto con il destinatario finale.
L'esperimento ha portato all'interessante risultato che fra due individui vi sono mediamente sei gradi di separazione, ovvero che la lettera era stata, mediamente, reinviata sei volte prima di raggiungere il destinatario finale.

Il Karate club di Zachary. W. W. Zachary è famoso per l'esperimento condotto all'interno di un club di Karate. Inizialmente è stato chiesto ad ogni atleta di compilare un questionario indicando quali fossero le persone con cui erano più legati. La Figura 1.2 ci mostra come ci fossero due principali persone, due leader, intorno alle quali si erano sviluppati due gruppi sociali abbastanza distinti. La cosa interessante è che, in seguito ad un litigio relativo alla tassa

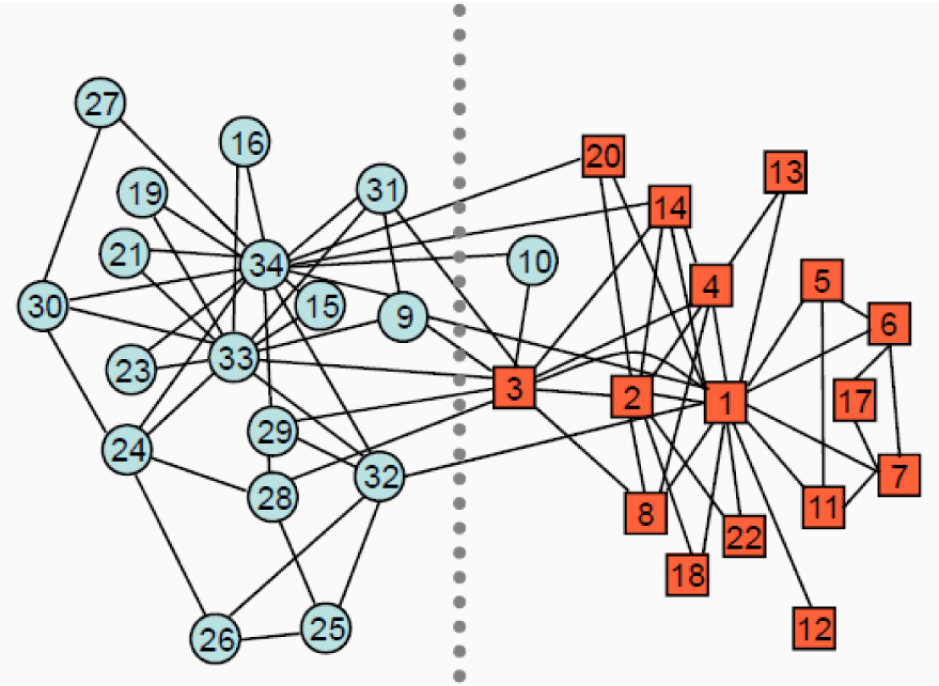


Figura 1.2: Il risultato dell'esperimento condotto da Zachary.

del dojo, si è effettuata una scissione del gruppo e i due sotto-gruppi formatisi in seguito alla discussione furono proprio quelli che si erano creati intorno ai due personaggi rilevanti (vediamo la scissione indicata dalla linea tratteggiata).

1.1.4 Le reti di informazioni

Vediamo ora alcuni esempi di reti di informazioni.

La struttura del web. Una prima analisi del web venne effettuata negli anni 2000. Ne vediamo il risultato nella Figura 1.3. Notiamo come si abbia un nucleo centrale, detto *SCC* (strongly connected component) che rappresenta quelle pagine web che portano un link alla pagina dalla quale sono state linkate. Sono quindi definite componenti fortemente connesse poiché permettono una navigazione continua fra di esse, potendo sempre tornare indietro.
Vi sono poi pagine (rappresentate dall'ala destra) *IN* che portano alla parte

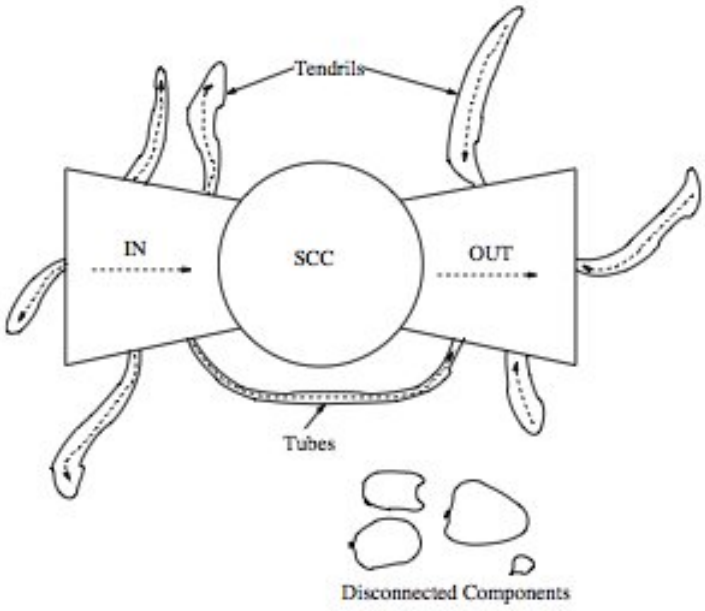


Figura 1.3: Il web ha la forma di un papillon.

SCC ma che non permettono di tornare indietro. Analogamente abbiamo pagine *OUT* che invece portano a pagine foglia che non hanno più link al loro interno.
Vi sono poi alcuni tentatacoli e tubi intorno alla struttura a papillon oltre che alcune componenti totalmente disconnesse.
Questo tipo di studio è stato fondamentale per lo sviluppo dei crawler web, primo su tutti il mastodontico Google, che sviluppa i suoi primi algoritmi basandosi proprio su questo tema (riuscendo così a soppiantare Altervista, precedente titano fra i motori di ricerca).

La rete delle citazioni scientifiche. Altro esempio di rete di informazioni è quello delle citazioni scientifiche. Ne vediamo una sezione nella Figura 1.4, dove abbiamo anche una visualizzazione che ci aiuta a capire quali sono i nodi più citati.

Il principio dell'influenza

Da quest'ultimo esempio possiamo definire l'importante principio dell'influenza: un attore sociale influente potrà determinare strategie di mercato, di politica, ecc. Nel caso delle citazioni scientifiche, un attore influente può essere utile per far circolare più velocemente le idee.

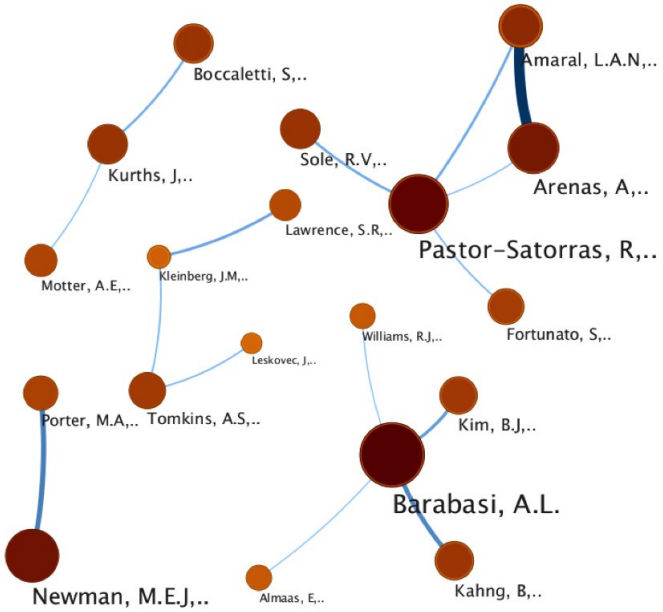
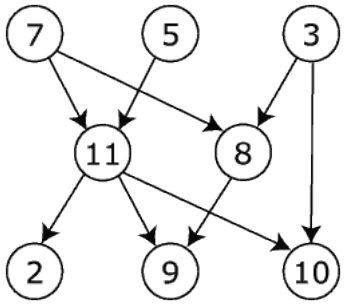


Figura 1.4: Parte della rete delle citazioni scientifiche.

Abbiamo un chiaro esempio di questo nel portale *slashdot*[2], un sito che si occupa di recensioni di siti web. Essendo slashdot molto seguito (è un nodo influente), ogni qual volta recensisce un certo portale, quest'ultimo sarà bombardato di visite (tanto da creare in alcuni casi un crash del portale stesso!). Tale fenomeno è così frequente che è possibile sentire frasi come —*I have been slashdotted*.

1.2 Richiami della teoria dei grafi

Come abbiamo già accennato, una rete complessa è facilmente rappresentabile come un grafo. È quindi necessario introdurre alcune nozioni sulla teoria dei grafi che ci saranno indispensabili per effettuare network analysis.



Un grafo $G = (V, E)$ è un insieme di vertici $V = v_1, v_2, ..., v_n$ e di archi (edges) $E = e_1, e_2, ..., e_n$. Tale definizione è quella delineata nella la teoria dei grafi. A seconda del contesto chiameremo i vertici in modo diverso (nodi in informatica, site in fisica, ecc.) e così gli archi (link in informatica, bond in fisica, ecc.)

[2] www.slashdot.com

1.2.1 Tipi di rete

Facciamo ora una rapida panoramica di tutte le tipologie di reti esistenti.

- **diretta/non diretta**: in una rete diretta gli archi sono direzionati. Un esempio di rete diretta è di Twitter, dove si può essere follower di qualcuno che non è tuo follower a sua volta: questa informazione è rappresentata dalla direzione della freccia. Un esempio di rete indiretta è invece Facebook dove ogni amicizia è necessariamente ricambiata e quindi gli archi non ha senso che siano direzionati poiché avranno sempre un contro arco nella direzione opposta.

- **pesata/non pesata**: in una rete pesata gli archi dispongono del concetto di peso, ovvero un valore che indica l'intensità della connessione.

- **multigrafo**: un multigrafo è un grafo che permette self-loop.

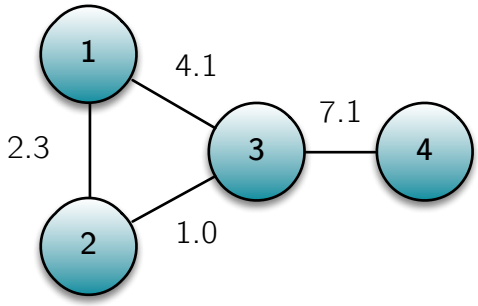
- **unipartita/multipartita**: una rete unipartita è una rete in cui tutti i nodi di un grafo sono assimilabili ad un gruppo soltanto (un grafo monotematico). Una rete multipartita è sostanzialmente la sovrapposizione di più grafi differenti, ovvero nodi di un certo gruppo possono essere collegati solo a nodi dello stesso gruppo.

- **statica/non statica (in evoluzione)**: la definizione è in merito alla topologia della rete. Una rete statica avrà sempre lo stesso numero di nodi e di archi, non modificherà quindi la sua topologia nel tempo. Una rete non statica vedrà invece spesso l'aggiunta di nuovi nodi e la rimozione di vecchi. Senz'altro la rete delle amicizie di Facebook è non statica.

- **dinamica/non dinamica**: la definizione è in merito alle proprietà dei nodi. Pensiamo ad una rete che studia la propagazione di una certa epidemia: un nodo può cambiare il suo stato da non infetto ad infetto. Spesso una rete dinamica sarà anche non statica ma le due definizioni sono comunque differenti fra loro.

1.2.2 Rappresentazione di una rete

Un grafo può essere rappresentato in diversi modi grazie all'uso di matrici. Consideriamo per esempio questo grafo:



Vediamone tre differenti rappresentazioni:

$$\begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \qquad \begin{pmatrix} 0 & 2.3 & 4.3 & 0 \\ 2.3 & 0 & 1.0 & 0 \\ 4.3 & 1.0 & 0 & 7.1 \\ 0 & 0 & 7.1 & 0 \end{pmatrix} \qquad \begin{pmatrix} 2 & -1 & -1 & 0 \\ -1 & 2 & -1 & 0 \\ -1 & -1 & 3 & -1 \\ 0 & 0 & -1 & 1 \end{pmatrix}$$

Matrice di adiacenza (A) Matrice di pesi (W) Matrice Lapaliciana (L)

- Nella *matrice di adiacenza* si ha che $A_{ij} = 1$ se esiste un collegamento fra i e j.
- Nella *matrice di pesi* si ha che $W_{ij} = x$ dove x è il peso della connessione fra i e j.
- La *matrice Laciana* è ottenuta come $L = D - A$ dove D è una matrice diagonale (costituita come: $d_{ii} = \sum_j A_{ij}$). Pertanto, le righe della Laplaciana saranno a somma zero.

1.2.3 Caratterisiche di una rete

Vediamo alcune variabili che possiamo definire riguardo una rete.

Cammino minimo fra due nodi (d_{ij})

Il cammino minimo d_{ij} tra i nodi i e j corrisponde alla distanza (pesata qualora ci siano pesi) minima fra tutti i possibili cammini tra i due nodi i e j. È anche chiamata distanza geodetica.

Cammino minimo medio di un grafo (l)

È possibile definire il cammino minimo medio come $l = \langle d_{ij} \rangle = \frac{1}{n(n-1)} \sum_{i \neq j} d_{ij}$.

Si considerano tutti i nodi del grafo e si calcola il cammino minimo fra ogni

coppia di essi (d_{ij} nella sommatoria). Dopodiché si divide per tutte le combinazioni di cammini possibili fra gli n nodi del grafo ($n(n-1)$). Chiaramente non vengono considerati i self-edges.

Diametro di un grafo (D)

Il diametro di un grafo è definito come $D = max(d_{ij})$ ovvero come il massimo fra i cammini minimi di tutti i nodi del grafo.

Componente di un nodo

È definito *componente* l'insieme dei nodi raggiungibili da un singolo nodo.

Grado di un nodo (k_i)

Il grado di un certo nodo i della rete è definito come $k_i = \sum_{j=1}^{n} A_{ij}$, ovvero il numero di connessioni di un nodo. Nel caso in cui il grafo sia diretto si avranno un k^{out} e un k^{in} per discriminare rispettivamente gli archi uscenti e quelli entranti.

Strenght di un nodo (s_i)

Viene definita come strenght di un nodo la variabile $s_i = \sum_{j=1}^{n} W_{ij}$, ovvero la somma dei pesi delle connessioni di un nodo.

Betweenness di un nodo (b_i)

La betweenness di un nodo è definita come $b_i = \sum_{j,k \in N, j \neq k}^{n} \frac{n_{jk}(i)}{n_{jk}}$, ovvero il numero di cammini minimi che passano dal nodo i-esimo (o da un collegamento tra due nodi).

Grado e numero di collegamenti

Il numero di archi di un grafo è definito in funzione del grado medio del grafo. Infatti ogni arco collega due nodi, pertanto con m archi vi saranno $2m$ nodi collegati da archi (eventualmente ripetuti). Tale numero, sarà uguale alla somma dei gradi di ogni singolo nodo (diviso due per non contarli due volte):

$$2m = \sum_{i=1}^{n} k_i \longrightarrow m = \frac{1}{2} \sum_{i=1}^{n} k_i = \frac{1}{2} \sum_{i,j} A_{ij}$$

Abbiamo poi che il grado medio di un grafo è $c = \frac{1}{n}\sum_{i=1}^{n} k_i$. Se combiniamo le due cose otteniamo $c = \frac{2m}{n}$.

Il massimo numero di archi (senza multi-edge e self-loop) è definito come $\binom{n}{2} = \frac{1}{2}n(n-1)$.

La densità di un grafo è quindi $\rho = \frac{m}{\binom{n}{2}} = \frac{2m}{n(n-1)} = \frac{c}{n-1}$.

Dunque ρ varia nel range $0 \leq \rho \leq 1$ e $\approx \rho = \frac{c}{n}$.

Una rete con ρ tendente a una costante con $n \to \infty$ sarà detta **densa** (poiché abbiamo un numero costante di archi all'aumento dei vertici), mentre una rete con $\rho \to 0$ per $n \to \infty$ sarà detta **sparsa** (il numero di archi tende a 0 all'aumentare dei vertici).

Si noti che questo genere di considerazioni non si possono fare puntualmente bensì sono effettubabili nel tempo e quindi con la crescita della rete.

Possiamo osservare che è improbabile che il numero di legami di amicizia raddoppi al raddoppiare della dimesione della popolazione, dunque il numero di amicizie è funzione del tempo dedicato a coltivare le relazioni di amicizia. Le reti di amicizia sono quindi sparse, così come la maggior parte delle reti. Un'eccezione è rappresentata dalle *food web* ovvero le reti rappresentanti catene alimentari: tali reti sembrano essere tendenzialmente dense.

Grafi regolari

I grafi regolari sono grafi con grado costante: ogni nodo avrà un pari numero di archi, perciò avremo una sorta di sfera di vertici collegati fra loro.

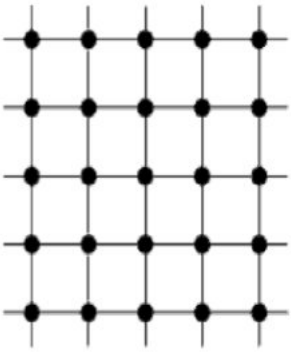


Figura 1.5: Un grafo con grado costante.

Motif di una rete

Sono detti *network motif* dei particolari sotto-grafi che spesso ricorrono all'interno di un network. Lo studio di motif può permettere di capire alcune caratteristiche peculiari della rete, oltre che a fornire indizi su come l'informazione venga processata.

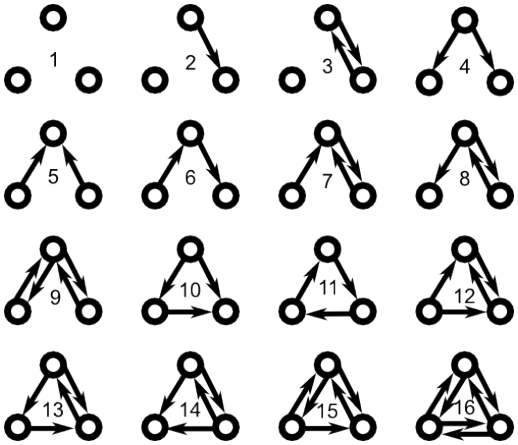


Figura 1.6: Tutti i possibili motifs generabili con tre nodi.

Coefficiente di clustering di un nodo (C_i)

Il coefficiente di clustering di un nodo i è calcolato come $C_i = \frac{2E}{k_i(k_i-1)}$ dove E è il numero di connessioni esistenti con i suoi vicini. Si ha quindi che il numero di connessioni esistenti con i suoi vicini è diviso da tutte le combinazioni possibili di connessioni con i suoi vicini. In sostanza si guarda quanto un nodo è effettivamente connesso rispetto a quanto potrebbe esserlo.

Coefficiente di clustering di un grafo (C)

Si può definire in due modi diversi:

- Watts & Strogatz definiscono C come la media di tutti i C_i.
- Neman definisce C come *(numero di triangoli)x3 / numero di triple connesse* (anche detto *transitività*).

Le due misure hanno valori differenti ma comportamenti analoghi.

Capitolo 2

Strumenti di analisi delle reti

2.1 Un tour fra i tool

Chiaramente trattare con le reti non è affare da carta e penna: gestire grandi quantità di dati è una delle cose che i software sanno fare meglio. Vediamo nella tabella una lista di tool utilizzabili per rappresentare ed elaborare reti.

Nome	**Descrizione**
Pajek	Analisi interattiva di social network e visualizzazione
Net Workbench	Analisi interattiva di network e visualizzazione
Netminer	Analisi interattiva di social network e visualizzazione
InFlow	Analisi interattiva di social network e visualizzazione
UCINET	Analisi interattiva di social network
yEd	Visualizzazione interattiva
Visone	Visualizzazione interattiva
Graphviz	Visualizzazione
NetworkX	Analisi interattiva di network e librerie Python
JUNG	Librerie Java per analisi di network e visualizzazione
igraph	Librerie C/R/Python per analisi di network
GTL	Librerie C++ per analisi di network
LEDA/AGD	Librerie C++ per analisi di network
Gephi	Analisi interattiva di network e visualizzazione

Nel nostro caso useremo **R**, un tool statistico gratuito e multi-piattaforma. Si tratta di un vero e proprio linguaggio di programmazione interpretato che offe diverse librerie personalizzate (anche molto sofisticate) per la visualizzazione. La community più attiva in questo senso è quella di **Gephi**.
Useremo, inoltre, il tool per analisi di network **igraph**.

2.2 Ghepi

Il *formato nativo* di Ghephi è il gexf, un file che elenca prima i nodi e poi le relazioni fra loro.
Al momento dell'import è possibile specificare la tipologia di grafo (diretto/non diretto), dopodiché è possibile agire sul pannello a sinistra per impostare un *layout* (si suggerisce Force Atlas che simula la gravità fra i nodi). L'uso di un layout appropriato rende la rete più leggibile, quindi, ci fornisce informazioni dirette senza effettuare alcuna analisi complessa.
Il pannello a sinistra in alto (*Ranking*) ci permette di evidenziare i nodi (sia intermini di colori che di grandezza) basandosi su diversi parametri (grado del nodo, grado di entrata, grado di uscita, ecc.).
Clickando sulla T in basso è possibile visualizzare il nome dei nodi.
In alto troviamo il pannello *Data Laboratory* dove possiamo interrogare il data set e ottenere informazioni in merito ai singoli nodi (oltre che farceli indicare sulla mappa con tasto destro -> View in Overview).
La tab *Preview* infine ci permette di esportare il nostro file.
Vediamo un esempio di export nella Figura 2.1.

Figura 2.1: Esempio di export in Gephi.

2.3 R e igraph

Introduciamo ora alcuni comandi per R.

2.3.1 Installazione di igraph

Per installare *igraph* eseguire:

```
> install.packages("igraph")
```

Caricare l'ambiente

Per caricare il pacchetto installato (va fatto ad ogni avvio di R):

```
> library("igraph")
```

Creare grafi

Crea grafi con caratteristiche topologiche diverse:

```
> g <- graph( c(0,1, 1,2, 3,4, 5,6) )
> e <- graph.empty(n=10)
> f <- graph.full(3, directed=TRUE)
> l <- graph.lattice(length=5, dim = 2)
> t <- graph.tree(n=10,children=2, mode="undirected")
> r <- graph.ring(n=11)
```

Importare dati CSV

```
> dat <- read.csv("path/simpledataset.txt")
> gf <- graph.data.frame(dat)
```

2.3.2 Stampare e visualizzare grafi

Stampa i dati fondamentali del grafo (in `gf` è definito un grafo):

```
> gf
Vertices: 3 Edges: 2 Directed: TRUE Edges:
[0] 'John' -> 'Alex' [1] 'Maria'-> 'Alex'

> plot(gf) //stampa il grafo
```

2.3.3 Estrazione di singole proprietà

Stampa dei dati relativi ai vertici:

```
> V(gf)
Vertex sequence:
[1] "John" "Maria" "Alex"
```

Stampa dei dati relativi agli archi:

```
> E(gf)
Edge sequence:
[0] John -> A
```

```
> E(gf)[weight > 3] //condizionale
Edge sequence:
```

Altri dati interessanti ($x definisce l'informazione sulla proprietà x):

```
> lenght(V(gf)) // qt di nodi
[1] 3
> V(gf)$name // label assegnati ai nodi
[1] "John" "Maria" "Alex"
> E(gf)$weight //peso degli archi
[1] 1 1
```

2.3.4 Grado dei nodi del grafo

Stampa di dati relativi ai gradi (del grafo, in uscita, del vertice `Alex`):

```
> degree(gf)
[1] 1 1 2
> degree(gf, mode="out")
[1] 1 1 0
> degree(gf,v="Alex")
[1] 2
```

2.3.5 Creare altre strutture

Crea un grafo a partire da un altro grafo:

```
> gu <- as.undirected(gf)
```

Crea una matrice `matrix(data, columns, row)`;
Nota le matrici in R sono definite per colonna prima e per riga dopo:

```
> M <- matrix(c(0,1,2,2,1,2,2,3),4,2)
> M
     [,1] [,2]
[1,] 0    1
[2,] 1    2
[3,] 2    2
[4,] 2    3
```

Crea un grafo a partire da una matrice (dobbiamo farne la trasposta, usiamo $t(M)$).

```
> g <- graph(t(M))
> g
Vertices: 4
Edges: 4
```

```
Directed: TRUE Edges:
[0] 0 -> 1
[1] 1 -> 2
[2] 2 -> 2
[3] 2 -> 3
```

2.3.6 Creare da matrice di adiacenza

```
> adj <- matrix(c(0,1,1,0,1,1,0,0,1,0,0,0,1,1,1,0),4,4)
> adj
[,1] [,2] [,3] [,4]
[1,] 0 1 1 1
[2,] 1 1 0 1
[3,] 1 0 0 1
[4,] 0 0 0 0
> graph.adjacency(adj)
Vertices: 4
Edges: 8
Directed: TRUE
Edges:
[0] 0 -> 1
[1] 0 -> 2
[2] 0 -> 3
[3] 1 -> 0
[4] 1 -> 1
[5] 1 -> 3
[6] 2 -> 0
[7] 2 -> 3
```

2.3.7 Attributi

Grafi, vertici e archi possono avere *attributi*; alcuni sono nativi (ad esempio width e name) mentre è sempre possibile aggiungerne di altri. Sono utili per effettuare *plot*.

```
> E(gw)$color <- "SkyBlue2"
> E(gw)$width <- 1
> E(gw)[weight > 3]$color <- "red"
> E(gw)[weight > 3]$width <- 2
```

Per conoscere gli attributi:

```
> list.edge.attributes
> list.vertex.attributes
```

```
> list.graph.attributes
```

2.3.8 Visualizzazione con plot e (tkplot)

Abbiamo due modi di stampare un grafo. Tramite la funzione `plot` abbiamo il grafo in 2D (statico), ma volendo possiamo anche adoperare `tkplot` che invece fornisce un grafico in 2D interattivo.

```
> plot.igraph(gw, layout=layout.kamada.kawai,
  edge.color = E(gw)$color,
  edge.width=E(gw)$width,
  vertex.color=V(gw)$color,
  vertex.label=V(gw)$name,
  vertex.size=3)
```

2.3.9 Layout dei plot

È possibile avere diffenti layout per il plotting dei grafi:

```
layout.random(graph, params, dim=2)
layout.circle(graph, params)
layout.sphere(graph, params)
layout.fruchterman.reingold(graph, ..., dim=2,
  verbose=igraph.par("verbose"), params)
layout.kamada.kawai(graph, ..., dim=2,
  verbose=igraph.par("verbose"), params)
layout.spring(graph, ..., params)
layout.reingold.tilford(graph, ..., params)
layout.fruchterman.reingold.grid(graph, ...,
  verbose=igraph.par("verbose", params)
layout.lgl(graph, ..., params)
layout.graphopt(graph, ..., verbose = igraph.par("verbose"),
  params = list())
  layout.mds(graph, d=shortest.paths(graph), ...)
layout.svd(graph, d=shortest.paths(graph), ...)
layout.norm(layout, xmin = NULL, xmax = NULL, ymin = NULL,
  ymax = NULL, zmin = NULL, zmax = NULL)
```

Possiamo distinguere in due categorie tali layout.

Layout base

- *layout.random*, distribuisce i vertici in modo casuale su un quadrato. Non ha parametri.

- *layout.circle*, distribuisce i vertici su un cerchio in modo equidistante. Non ha parametri.

- *layout.sphere*, distribuisce i vertici in modo (approssimativamente) uniforme sulla superfice di una sfera. È un layout 3D.

Layout force-based

- *layout.fruchterman.reingold*, algoritmo proposto da Reingold e Fruchterman. Per i parametri vedere la documentazione.

- *layout.kamada.kawai*

- *layout.spring*

2.3.10 Reti egocentriche

Una *rete egocentrica* è una proiezione della rete su un individuo specifico di tale rete.

Come costruirla

```
// Definiamo il vertice "ego"
> nodes <- V(gw)["miriam01"]
// Focalizziamo i nodi attorno ad esso
> nodes <- c(nodes, neighbors(gw,"miriam01"))
// Prendiamo il sottografo di interesse
> subgw <- subgraph(gw,nodes)
```

2.3.11 Altre funzionalità di R

Vediamo gli altri strumenti forniti da R che potrebbero comunque rivelarsi utili.

2.3.12 Istruzioni base

Funzione	Significato
`q()`	quit
`help(funzione)`	fornisce info sulla funzione
`example(argomento)`	fornisce esempi sull'argomento
`source("filename.R")`	carica le istruzioni da file
`x <- espressione`	assegna ad `x` il risultato dell'espressione
`c(...)`	crea collezioni di dati (vettori)
`is.na(x)`	restituisce un vettore di valori TRUE e FALSE per ogni elemento di `x` che è not available
`is.nan(x)`	restituisce un vettore di valori TRUE e FALSE per ogni elemento di `x` che è not a number
`funzione(x)`	con `x` vettore, la funzione verrà applicata iterativamente ad ogni elemento di `x`

Esempio di operazioni su elementi vettori:

```
> x <- c(1000,100,10,1)
> 1/x
[1] 0.001 0.010 0.100 1.000
```

Esempio di concatenazione di vettori:

```
> y <- c(x,0,x)
> y
[1]1000 100 10 1 01000 100 10 1
```

Esempio di aritmetica vettoriale:

```
> x <- c(1000,100,10,1) > y <- 1:4
> v <- 2*x + y + 1
> v
[1]2002203 24 7
```

2.3.13 Operatori aritmetici per vettori

Funzione	Significato
`+, -, *, /, ^`	operatori elementari
`log, exp, sin, cos, tan, sqrt`	funzioni aritmetiche
`max(x), min(x)`	restituiscono il massimo (o il minimo) valore di x
`range(x)`	restituisce `c(min(x),max(x))`
`length(x)`	numero di elementi in `x`
`sum(x), prod(x)`	somma e prodotto degli elementi di `x`
`sort(x)`	restituisce un vettore ordinato coi valori di `x`

Esempio d'uso:

```
> sqrt(-17)
[1] NaN
Warning message:
In sqrt(-17) : NaNs produced

> sqrt(-17+0i)
[1] 0+4.123106i
```

2.3.14 Media e varianza

Funzione	Espressione equivalente
mean(x)	sum(x)/length(x)
var(x)	sum((x-mean(x))^2/(length(x)-1)

Definire una nuova funzione

Definiamo ad esempio la funzione deviazione standard (radice quadrata della varianza):

```
> stderr <- function(x) sqrt(var(x)/length(x))
```

2.3.15 Espressioni regolari

Espressione regolare	Espressione equivalente
1:30	c(1, 2, 3, ..., 30)
30:1	c(30, 29, 28, ..., 2, 1)
seq(from, to, by, length)	crea un range da un valore ad un altro con un passo e una dimensione massima
seq(-5, 5, .2)	c(-5.0, -4.8, ..., 4.8, 5.0)
seq(length=51, from = -5, by = .2)	c(-5.0, -4.8, ..., 4.8, 5.0)
rep(x, times)	ripete il vettore x per un certo numero di volte
rep(x,each)	ripete ogni valore di x per un certo numero di volte

2.3.16 Vettori di stringhe

In alcuni casi, possiamo usare l'attributo *names* per identificare gli elementi:

```
> fruit <- c (5,10,1,20)
> names(fruit) <- c("orange", "banana", "apple", "peach")
```

```
> lunch <- fruit[c("apple","orange")]
> lunch
apple orange
1     5
```

Notiamo che si tratta di uan sorta di *dizionario*, infatti a `apple` ho associato 1 e ad `orange` ho associato 5.
Utile in connessione con data.frames, che è una funzione che crea una struttura costituita da tabelle dove è possibile associare dei nomi a dei valori.

2.3.17 Conversioni

Possiamo trasformare i caratteri in interi e viceversa:

```
> as.character(x) -> digits
> as.integer(digits) -> d
```

È poi possibile manipolare le stringhe tramite funzioni quali:

```
> strsplit(x, split, ...)
> paste(x,sep,collapse)
```

Array a più dimensioni

Possiamo lavorare direttamente con array e matrici; se supponiamo che `z` sia un vettore di 1500 elementi possiamo usare l'attributo `dim` come segue:

```
> dim(z) <- c(3,5,100)
```

Questa funzione ci consente di trattare il vettore come fosse un array multidimensionale 3x5x100 L'accesso ai singoli elementi avviene tramite indice:

```
> z[1,1,1], z[2,1,1], ...., z[3,5,100] etc..
```

Mentre l'accesso a porzioni dell'array è possibile in questo modo:

```
> z[2,,]
```

Matrici

È possibile definire una matrice in due modi:

- Usando la funzione `array` e l'attributo `dim`,
- Usando la funzione `matrix`.

```
> M <- array(1:20, dim=c(4,5))
> M
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    5    9   13   17
[2,]    2    6   10   14   18
[3,]    3    7   11   15   19
[4,]    4    8   12   16   20

> i <- array(c(1:3,3:1), dim=c(3,2))
> i
     [,1] [,2]
[1,]    1    3
[2,]    2    2
[3,]    3    1

> M[i]
[1] 9 6 3
```

2.3.18 Operazioni su matrici

Funzione	Significato
`t(M)`	genera la matrice trasposta di `M`
`nrow(M), ncol(M)`	numero di righe e colonne di `M`
`diag(M)`	restituisce un vettore che è la diagonale di `M` (se usata su vettore restituisce una matrice che ha quel vettore come diagonale)
`b <- M * x`	prodotto di `M` con un vettore `x`
`solve(M,b)`	risolve il sistema lineare (`x = M-1b`)
`solve(M)`	inversa di `M` (M^{-1})

Autovalori ed autovettori

È possibile calcolare gli autovettori e gli autovalori di una matrice simmetrica Sm tramite `eigen(Sm)` :

```
> Sm <- matrix(c(1,1,1,0),2,2)
> Sm
  [,1] [,2]
[1,] 1 1
[2,] 1 0
> ev <- eigen(Sm)
> ev$vec
  [,1]     [,2]
```

```
[1,] -0.8506508  0.5257311
[2,] -0.5257311 -0.8506508
> ev$val
[1]  1.618034 -0.618034
```

Capitolo 3

Misurare la struttura di una rete

Quando studiamo una rete siamo interessati nel carpire le informazioni fondamentali da essa rappresentata e per fare questo utilizziamo le *misure di centralità*.

3.1 Misure di centralità fondamentali

Cerchiamo i nodi più importanti. Ma che cosa si intende con 'importante'? È forse tra i primi a ricevere un'informazione, oppure è essenziale per la propagazione delle informazioni nel contesto della rete, oppure ancora è in grado di influenzare facilmente gli altri nodi della rete?

3.1.1 Degree centrality

È la misura più intuitiva che possiamo immaginare ed è in grado di misurare la *popolarità* di un certo nodo. Semplicemente contiamo il numero di archi entranti o uscenti (nel caso di grafo diretto) o genericamente incidenti (nel caso di grafo non diretto).

$$k_i = \sum_{j=1}^{n} A_{ij} \tag{3.1}$$

Uso nelle citazioni scientifiche

All'interno del grafo delle citazioni scientifiche se ne fa un uso particolare, nella fattispecie si definisce l'*h-index* che è calcolato a partire dall'in-degree di un nodo (un articolo). L'h-index è riferito ad un autore di h articoli tali che $k_i^{in} \geq h$, ovviamente con l'h massimo della rete (in sostanza l'articolo più citato di tutti).

3.1.2 Closeness centrality

La *closeness centrality* misura la distanza geodetica media di un nodo da tutti gli altri nella rete.

$$k_i = \frac{1}{n} \sum_j d_{ij} \tag{3.2}$$

Attenzione! Più **basso** è il valore di centralità, più il nodo sarà centrale. Questo tipo di misura è molto utile per capire quali nodi sono in grado di trasmettere informazioni più velocemente rispetto agli altri.

Varianti e problemi

Talvolta si tende ad escludere nel calcolo il nodo stesso (il comportamento della misura comunque non cambia di molto).

$$l_i = \frac{1}{n-1} \sum_{j(\neq i)} d_{ij}$$

Nel caso in cui si voglia ottenere una misura più intuitiva (la closeness aumenta con l'aumentare dell'effettiva centralità) è sufficiente utilizzare la formula inversa:

$$C_i = \frac{1}{l_i} = \frac{n-1}{\sum_{j(\neq i)} d_{ij}}$$

La closeness centrality soffre quindi di alcuni problemi poiché i valori delle distanze tendono ad essere molto piccoli. Le distanze crescono logaritmicamente e quindi i valori possibili di l_i hanno una variazione minima. Questo rende difficile effettuare raffronti fra nodi di centralità simili; basti pensare che nella rete degli attori quello con closeness centrality maggiore è Chistopher Lee ($l_i = 2.4138$) ed il meno centrale è Leia Zanganeh (l_i= 8.6681). Gli altri attori (più di un milione e mezzo di individui) si orientano in questo piccolo range, ad esempio il secondo classificato (Donald Pleasence) si trova a $l_i = 2.4164$: la differenza è davvero millesimale.

Un altra problematica si riscontra nel caso di reti suddivise in più componenti non connesse: i nodi che si trovano in componenti più piccole avranno automaticamente centralità più alta, ma questo non è corretto (non si tiene conto delle altre componenti).

È eventualmente possibile usare la *media armonica*, una soluzione elegante ma poco usata:

$$C'_i = \frac{1}{n-1} \sum_{(j \neq i)} \frac{1}{d_{ij}}$$

La media armonica è poco usatap perché il problema delle multi-componenti è poco rilevante, tendenzialmente si svolge l'analisi sulla componente più importante e massiccia della rete, da ciò, possiamo ignorare i risultati sulle altre componenti.

3.1.3 Betweenness centrality

La *betweenness centrality* è una misura calcolabile sia a livello di nodo che a livello di arco ed esprime la possibilità che un nodo (o un arco) appaia lungo il cammino minimo tra due altri vertici.
Siamo in sostanza interessati a capire quali fra i nodi del grafo sono quelli che connettono diverse componenti e quindi sono rilevanti per la trasmissione di informazione oltre che per la connettività generale di parti differenti della medesima rete.
La rimozione di un nodo che ha alta betweenness centrality porterebbe a grossi danni, possibilimente alla sconnessione di parti significative della rete.
Ma nel mondo reale abbiamo che non tutti i vertici scambiano informazioni con la stessa frequenza, i messaggi non necessariamente seguono cammini casuali ed inoltre i percorsi scelti dai messaggi non sono necessariamente minimi. Questi tre fenomeni fanno si che i nodi con alta betweenness non siano così fondamentali e nonostante questa possa sembrare una cattiva notizia in realtà dovremmo essere sollevati: una rete che non si fonda solo sui nodi con alta betweenness è una rete *più robusta*.

Dal punto di vista matematico definiamo la betweenness in questo modo (pensiamo ad una rete non diretta e fortemente connessa):
Dato n_{st}^i che vale 1 se dal nodo i passa almeno un cammino minimo da s a t e 0 altrimenti, la betweenness del nodo i è

$$x_i = \sum_{st} n_{st}^i \tag{3.3}$$

Calcolata per ogni nodo s ed ogni nodo t del grafo.

Bridges e hub

È importante non confondere i concetti di betweenness e degree. Tramite la misura di degree centrality definiamo i cosiddetti *hub* cioè quei nodi che hanno molti archi. Invece i nodi con alta betweenness centrality sono detti *bridges* poiché fungono da collegamento fra due parti di rete. Più precisamente, i *bridges* sono gli archi che effettuano le connessioni e il nodo con alta betweenness è detto *broker*.

3.2 Misure di centralità più sofisticate

Lo studio delle reti ha portato ad una interessante riflessione: tutte le misure precedentemente considerate analizzano in maniera statica e democratica la rete ma in realtà sappiamo bene che tipicamente non tutti i nodi sono ugualmente importanti (ricordiamo lo *slashdot effect*). Le misure di *Eigenvector*

centrality e *Katz centrality* prendono in considerazione questo pensiero fornendo quindi una misura più realistica rispetto agli operatori statici. Tali misure vengono adoperate nell'ambito della *link analisys* che si occupa proprio di fare quanto detto: studiare una rete anche in relazione alle modalità con cui le informazioni vengono trasferite da un nodo all'altro (più che in relazione ai cammini minimi i quali spesso non sono neanche noti a priori).

3.2.1 Hubs and Authorities (HITS)

Vogliamo trovare un *buon sito* (*autorevole*) relativamente ad un certo argomento. Su cosa potremmo basarci? Intuitivamente un buon sito è un sito che è suggerito da *esperti* di cui ci fidiamo (concetto di *trust*). Pensiamo dunque ad un nodo esperto: un suo link ad un sito significherebbe un voto di positività a quest'ultimo.
Introduciamo quindi in ogni nodo due nuovi valori:

- *Hub*: valuta la qualità dell'esperto, cioè il numero totale di voti che ha assegnato. Ottimi esempi di hub sono liste di giornali, guide per lo studente, slashdot, ecc.
- *Authority*: valuta la qualità del contenuto di un certo sito (la quantità di voti ricevuti dagli esperti). Ottime autorità potrebbero essere home page di quotidiani, home pagine di corsi, ecc.

La misura è ottenuta in maniera *iterativa*.

L'algoritmo

Passo 1 Inizialmente vengono contati i link entranti come voti che valgono 1. Con i nodi arancioni rappresentiamo gli esperti (alto valore di hub) e con i nodi verdi le authorities (alto valore di authority).

Passo 2 Le votazioni precedenti influenzano il valore di hub degli esperti: il valore di hub di un esperto è la somma dei voti dei singoli siti cui l'hub si riferisce. Vediamo ad esempio che il secondo esperto a partire dall'altro ha valore di hub pari a undici poiché puntava a authority che nella totalità avevano $2 + 2 + 4 + 3 = 11$ voti.

Passo 3 Infine, si rieffettua il passo 1 ma ora i voti non valgono più 1 bensì il valore di hub dell'esperto che vota quella certa authority (Figura 3.3. Si avrà che l'authority totale di un nodo sarà la somma di tutti i valori di hub degli esperti che lo puntano.
Vediamo ad esempio che Amazon ha ricevuto un punteggio pari a 12 poiché puntato da esperti con valore di hub $3 + 3 + 6 = 12$.

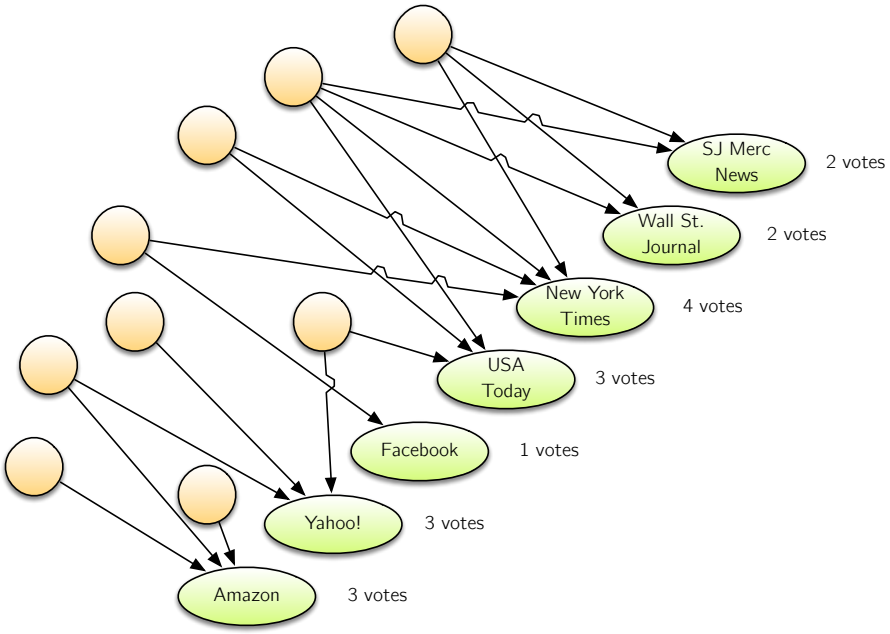


Figura 3.1: Contiamo i voti dai presunti esperti.

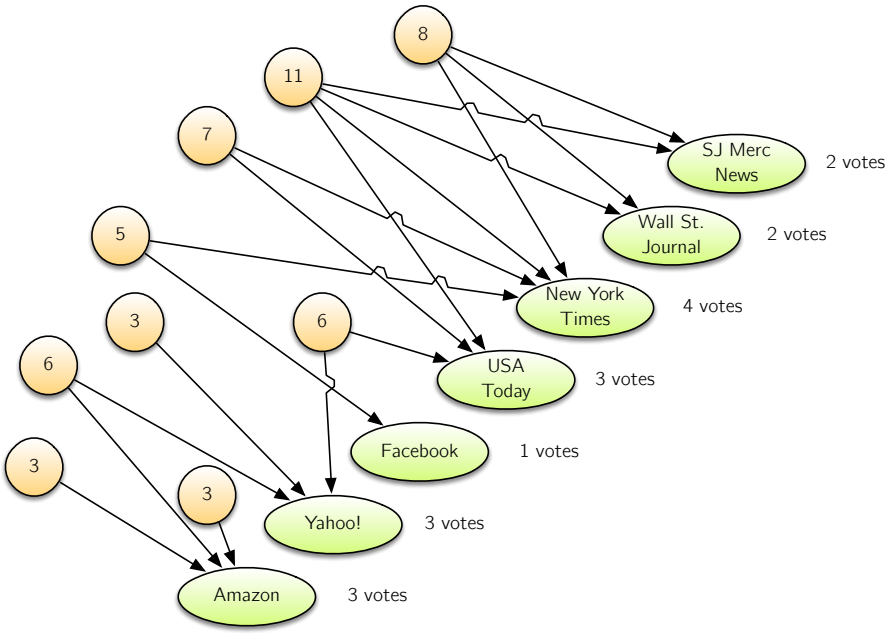


Figura 3.2: Definiamo un nuovo valore di hub per gli esperti.

La definizione è quindi ricorsiva, un buon hub conduce verso molte buone autorità e una buona autorità è collegata da molti buoni hub.

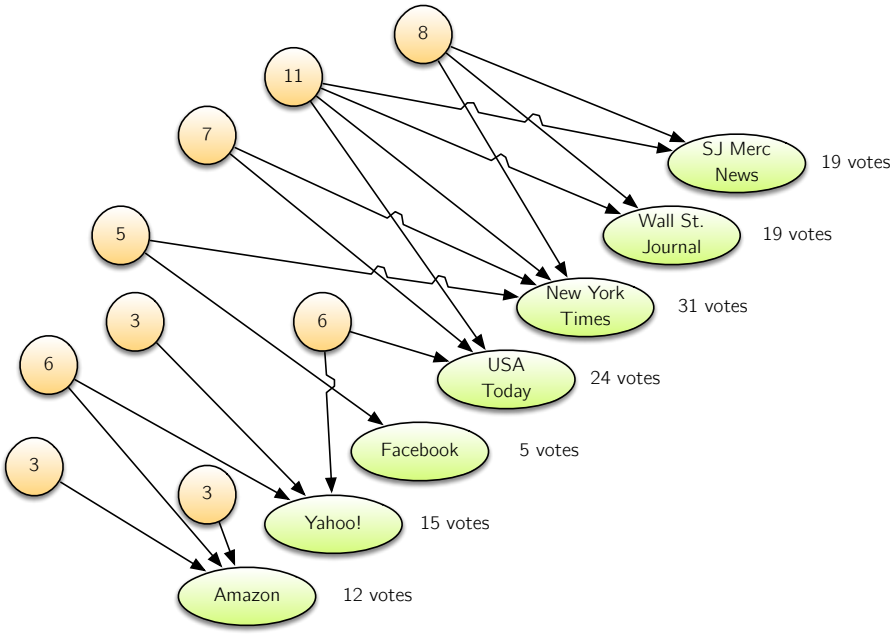
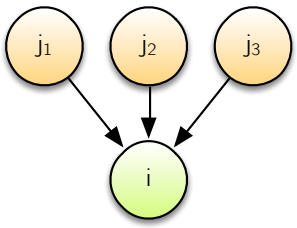


Figura 3.3: I valori di authority vengono aggiornati.

Definizione matematica

Definiamo con a_i il valore di authority di un nodo i e con h_i il suo valore di hub. Inizializziamo $a_j = 1, h_i = 1$.



Dobbiamo dunque iterare (con j i nodi che puntano ad i):

$$\forall i : a_i = \sum_j A_{ji} h_j$$

$$\forall j : h_i = \sum_j A_{ij} a_j$$

$$\forall i : \sum_i a_i = 1, \sum_j h_j = 1$$

Definizione matriciale

In termini di matrici possiamo definire $a = h = 1_n$ (elevato a n poiché è il valore di n per quanti nodi ci sono con archi entranti). Quindi possiamo ripetere:

$$h = Aa$$

$$a = A^T h$$

e dunque normalizzare.
Pertanto $a = A^T Aa$ e $h = AA^T h$. Dopo k passi del ciclo avremo: $a = (A^T A)^k a$ e $h = (AA^T)^k h$.

Autovalori ed autovettori

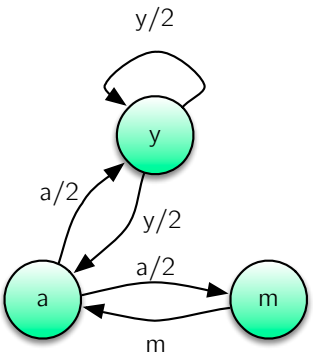
Dato un vettore x e una matrice M, sia $Mx = \lambda x$ per qualche scalare λ. Abbiamo che x è un *autovettore* e λ è il suo *autovalore*.
Pertanto osserviamo che se la matrice M è simmetrica, (cioè $A^T A$ e AA^T sono simmetriche) allora M ha n autovettori ortogonali $w_1, w_2, ..., w_n$ che formano una base (un sistema di coordinate) con autovalori $\lambda_1, \lambda_2, ..., \lambda_n$ dove vale sempre che $| \lambda_i | \geq | \lambda_{i+1} |$.

Si riesce a dimostrare che il vettore a delle autorità corrisponde all'autovettore di $A^T A$ che corrisponde all'autovalore maggiore λ_1. Analogamente accade con h e AA^T.

3.2.2 Page Rank

Altro interessante indice è il *page rank* che, intuitivamente, cerca di formalizzare il fatto che un voto proveniente da una pagina importante valga molto più di tanti voti da pagine non rilevanti (l'effetto slashdot).
Definiamo il rank r_i di un nodo i come:

$$r_i = \sum_j A_{ij} \frac{r_j}{k_j^{out}} \tag{3.4}$$



In sostanza il rank di un nodo è definito come la somma (per ogni nodo ad esso collegato) delle frazioni fra il rank del nodo collegato e il suo grado in uscita. Ad esempio gli archi uscenti da a valgono il rank di a fratto il grado di a (2). Si tratta quindi di risolvere un sistema per ottenere r_a, r_y ed r_m.

Il modello di Google

Google adopera un modello più complesso, che considera anche quei salti che avvengono via bookmark o barra degli URL. Secondo Google (negli anni 2000) la probabilità che un internauta acceda ad una pagina senza seguire gli URL di altre pagine (quindi tramite bookmark per esempio) è del 15%. Perciò la formula diventa (con β la probabilità di accesso non via bookmark):

$$r_i = (1-\beta)\sum_j A_{ij}\frac{r_j}{k_j^{out}} + \beta\frac{1}{n} \tag{3.5}$$

3.3 Gruppi di vertici

Abbiamo sempre esaminato i grafi considerandoli come assolutamente casuali ma in realtà sappiamo che in natura esistono delle ragioni per cui alcuni nodi sono legati o meno. In natura c'è ordine e questo ordine si riflette anche nella rappresentazione delle reti. Andiamo quindi a studiare alcune strutture elementari.

3.3.1 Clique

Una *clique* è un sottoinsieme massimale di vertici in un grafo non diretto completamente connesso.

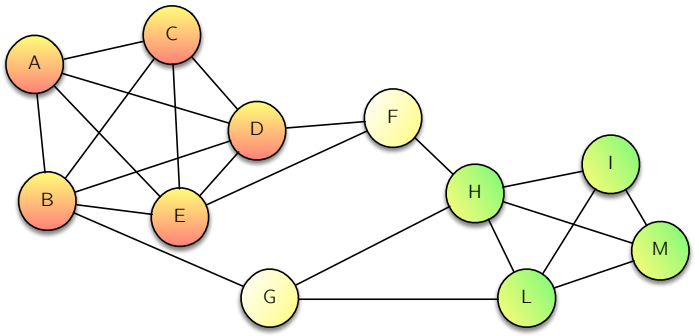


Figura 3.4: Vediamo due clique (una in rosso e una in verde).

In sostanza ogni nodo appartenente ad una clique è connesso con ogni altro nodo componente quella stessa clique.
Casualmente è difficile trovare clique ma invece in natura è più probabile: la presenza di clique indica la presenza di gruppi molto coesi.

3.3.2 K-plex

Un *k-plex* è una clique imperfetta, ovvero un k-plex di dimensione n è un sottografo massimale di n vertici tale che ogni elemento è connesso ad almeno

n-k altri elementi. Quindi esiste l'equivalenza clique = 1-plex poiché in una clique ogni nodo è connesso con tutti gli altri meno uno (egli stesso).

3.3.3 K-core

Il *k-core* è in sostanza la nozione inversa del k-plex: un k-core è un insieme massimale di vertici tale che ognuno di essi è connesso ad almeno k altri nel sottoinsieme. Pertanto un k-core = (n-k)-plex.
Sono più semplici da trovare (algoritmicamente parlando) dei k-plex e per questo sono molto utili.

3.4 Transitività aka coefficiente di clustering

È celebre la frase '*Un amico di un mio amico è un mio amico*' e sappiamo che tale affermazione è una rappresentazione del concetto di small word. Riscontriamo spesso una verità in questo concetto nelle reti sociali.
Vogliamo quindi studiare quelle situazioni in cui se a è amico di b e b è amico di c, c è amico di a ovvero le situazioni in cui si formano tre nodi collegati fra loro.

3.4.1 Definizioni di transitività

La prima e più semplice definizione si basa sulla ricerca di tutti i cammini di lunghezza 2 che si chiudono all'interno della rete. Si fa quindi un calcolo probabilistico classico (casi favorevoli su casi possibili) cioè:

$$C = \frac{\#\text{ cammini chiusi lunghi 2}}{\#\text{ cammini lunghi 2}} \tag{3.6}$$

Chiaramente questo valore si orienterà fra 0 ed 1 ed indicherà il livello di transitività (con 1 abbiamo una clique). Tale valore è noto anche come *coefficiente di clustering*.

Alternativamente è possibile calcolare la transitività in termini di triangoli. Tre nodi connessi fromano un triangolo che è visibile da tre punti di vista diversi (uno per nodo) e per ogni nodo è possibile avere due combinazioni diverse di cammini lunghi due, quindi abbiamo le due formule:

$$C = \frac{\#\text{ triangoli} \times 6}{\#\text{ cammini lunghi 2}} \tag{3.7}$$

$$C = \frac{\#\text{ triangoli} \times 3}{\#\text{ triple connesse}} \tag{3.8}$$

Definire una transitività alta

Come possiamo dire che un certo valore di transitività è alto?
Come in altri casi, si genera una rete casuale con lo stesso numero di vertici ed archi e si raffrontano i valori di C della rete originale e di C_{RAND} dalla rete casuale. La transitività sarà alta se $C >>> C_{RAND}$ (nel caso della rete degli attori $0.20 >>> 0.0003$)

3.4.2 Coefficiente di clustering locale

Può essere interessante calcolare il coefficiente di clustering a livello di un singolo nodo i.

$$C_i = \frac{\#\text{ coppie di vicini di } i \text{ connesse}}{\#\text{ coppie di vicini di } i} = \frac{2E_i}{k_i(k_i+1)} \tag{3.9}$$

Alternativamente si può considerare la transitività di un grafo facendo la media fra tutte le transitività singole dei nodi e quindi avere $C_{WS} = \frac{1}{n}\sum_{i=1}^{n} C_i$.

3.5 Reciprocità e segni sugli archi

Non solo i triangoli sono interessanti: potrebbe essere utile anche considerare quei casi in cui un arco è reciprocato (ovvero è presente in entrambe le direzioni).

3.5.1 Misurare la reciprocità

Sostanzialmente siamo interessati ai cicli di lunghezza due:

$$r = \frac{1}{m}\sum_{ij} A_{ij}A_{ji} = \frac{1}{m}TrA^2 \tag{3.10}$$

Si considera quindi la somma del numero di cicli di lunghezza due sul numero totale di archi presenti nel grafo.

3.5.2 Archi con segno

Possiamo anche attribuire agli archi un senso di *positivo* e *negativo*. Grazie a questa nozione, possiamo definire anche il concetto di equilibrio.
Si noti che il fatto che un arco fra due vertici non esista non significa che fra quei vertici vi sia un arco negativo. Vediamo in Figura 3.5 alcuni equilibri: la loro definizione non è così intuitiva come sembra.

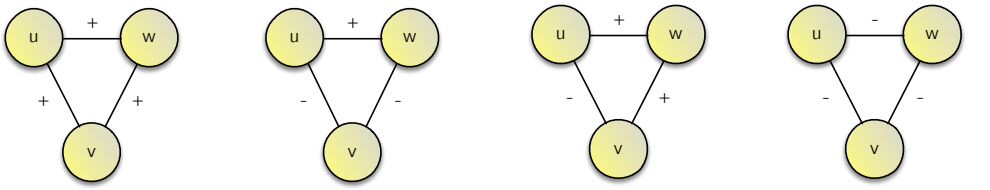


Figura 3.5: Esempi di equilibri.

Bilanciare una rete

Se riusciamo a trovare degli equilibri interni alla rete, possiamo far collassare gruppi di vertici in modo da poter studiare la rete più facilmente.

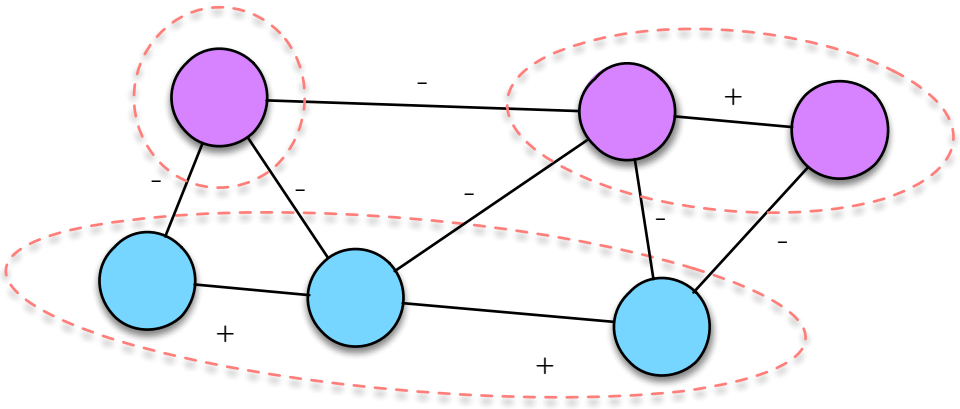


Figura 3.6: I nodi collassano secondo le linee tratteggiate.

3.6 Misure di similarità

È interessante capire quando due nodi sono simili, basti pensare all'enorme applicazione dei *recommendation systems*. Ci concentriamo ora sulle *equivalenze strutturali*.

3.6.1 Cosine similarity

Usiamo una metafora per considerare la similarità. Se i nosti nodi fossero vettori, il loro grado di similarità potrebbe essere rappresentato dal coseno che intrinsecamente ci fornisce una misura di somiglianza (in termini di angolo) fra

due vettori. In geometria abbiamo quindi che:

$$x \cdot y = \mid x \mid\mid y \mid \cos\theta$$

dove θ è l'angolo compreso fra il vettore x e il vettore y.
Pertanto considerando le righe (e colonne) i-esima e j-esima della matrice di adiacenza come due vettori, possiamo definire il coseno dell'angolo tra i due come la nostra misura di similarità σ_{ij}.

$$\sigma_{ij} = \cos\theta = \frac{x \cdot y}{\mid x \mid\mid y \mid} = \frac{\sum_k A_{ik}A_{kj}}{\sqrt{\sum_k A_{ik}^2}\sqrt{\sum_k A_{kj}^2}} \tag{3.11}$$

La formula è tale poiché ricordiamo che $\mid x \mid = \sqrt{x^2}$. Nel caso in cui la rete non sia pesata, allora $A_{ij}^2 = A_{ij}$ e quindi al denominatore possiamo ottenere $\sum_k A_{ik} = k_i$ cioè il degree del nodo. La nostra formula si semplifica quindi in:

$$\sigma_{ij} = \frac{\sum_k A_{ik}A_{kj}}{\sqrt{k_i k_j}} = \frac{n_{ij}}{\sqrt{k_i k_j}} \tag{3.12}$$

dove nij rappresenta il numero di nodi in comune fra i nodi i e j (a cui sono entrambi collegati).

Un esempio

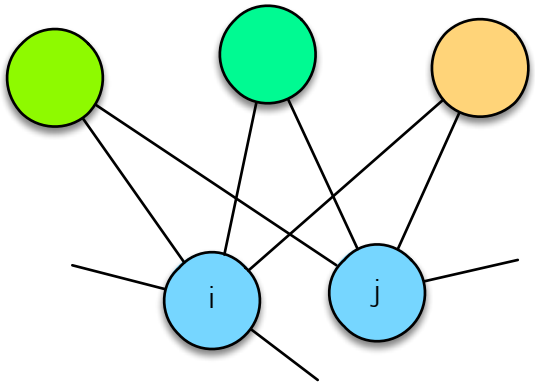


Figura 3.7: Calcoliamo la similarità fra i e j.

Abbiamo che:

$$\sigma_{ij} = \frac{n_{ij}}{\sqrt{k_i k_j}} = \frac{3}{\sqrt{5 \cdot 4}} = 0.671...$$

È quindi chiaro che con un valore 1 avremmo una situazione nella quale tutti i nodi vicini di due nodi sono in comune.

3.6.2 Coefficienti di Pearson

Un altro metodo per normalizzare il conteggio dei vicini in comune consta nel confronto fra il conteggio dei vicini nella rete effettiva e quello ottenuto in una rete in cui i vicini sono scelti casualmente. Come sappiamo possiamo contare i vicini tramite $n_{ij} = \sum_k A_{ik}A_{kj}$.
Nel grafo casuale (considerando i gradi k_i e k_j) avremo che il nodo i deve scegliere in tutto k_i nodi su un totale di $n-1$ (su grandi numeri approssimabile a n). Moltiplichiamo quindi questo valore per k_j, le scelte possibili per j. Otteniamo pertanto:

$$\frac{k_i k_j}{n}$$

Non ci resta che fare la differenza fra il valore effettivo ed il valore casuale, proprio come visto nella transitività:

$$\sum_k A_{ik}A_{kj} - \frac{k_i k_j}{n} = \sum_k A_{ik}A_{kj} - \frac{1}{n}\sum_k A_{ik}\sum_l A_{jl} =$$

$$\sum_k A_{ik}A_{kj} - n\langle A_i\rangle\langle A_l\rangle = \sum_k \left[A_{ik}A_{kj} - \langle A_i\rangle\langle A_l\rangle\right] =$$

$$\sum_k (A_{ik} - \langle A_i\rangle)(A_{kj} - \langle A_l\rangle) \tag{3.13}$$

Questa ultima formula è detta *covarianza* $cov(A_i, A_j)$ di due righe della matrice di adiacenza.

Normalizzazione della covarianza

I valori restituiti dalla covarianza possono essere molteplici e quindi essere difficilmente rapportabili. Si normalizza quindi la covarianza per avere valori compresi fra -1 e 1.
Il massimo valore ottenibile dalla covarianza tra due vettori corrisponde a quando essi sono esattamente uguali che equivale alla varianza di entrambi gli insiemi σ_i^2 oppure σ_j^2. Nel caso simmetrico abbiamo invece $o_i \cdot o_j$. Useremo quindi questo valore per rappresentare la massima covarianza; normalizziamo sulla variabile r_{ij} in questo modo:

$$r_{ij} = \frac{cov(A_i, A_j)}{\sigma_i \sigma_j} = \frac{\sum_k (A_{ik} - \langle A_i\rangle)(A_{kj} - \langle A_l\rangle)}{\sqrt{\sum_k (A_{ik} - \langle A_i\rangle)^2}\sqrt{\sum_k (A_{kj} - \langle A_l\rangle)^2}} \tag{3.14}$$

Misure alternative

Invece di sottrarre il numero di archi attesi nel caso random, potremmo pensare di metterlo a denominatore:

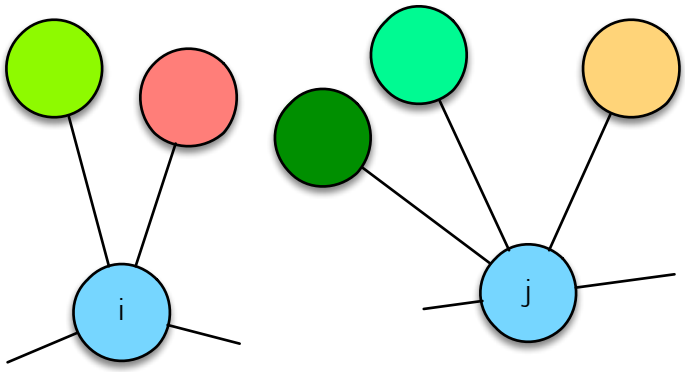
$$\frac{\sum_k A_{ik}A_{kj}}{\frac{k_i k_j}{n}} = n\frac{\sum_k A_{ik}A_{kj}}{\sum_k A_{ik}\sum_k A_{kj}} \tag{3.15}$$

Inoltre possiamo definire la *distanza euclidea* (che sarebbe poi da normalizzare ulteriormente):

$$d_{ij} = \sum_k (A_{ik} - A_{jk})^2 \tag{3.16}$$

3.6.3 Equivalenze regolari

Possiamo anche studiare le similarità fra due nodi vicini (i e j) anche se questi non hanno nodi vicini in comune.



Si svolgono misure che coinvolgono analisi spettrale (si lavora con autovettori e autovalori). I risultati di tali analisi sono spesso difficli da comprendere.

3.7 Homophily e Assortative Mixing

Parliamo ora di somiglianza fra nodi e di come tali nodi possano tendere ad essere più o meno coesi in base a questa caratteristica. Possiamo ovviamente avere diversi livelli di similarità: ad esempio basata sul sesso, sulla razza, sul genere, ecc.
Sorprendentemente possiamo avere comportamenti molto diversi, ad esempio nodi con grado simile potrebbero essere molto legati ma anche estremamente distaccati, dipende un po' dalla tipologia della rete.
Questo fenomeno si chiama *omofilia* e secondo Newman è possibile studiarlo tramite l'Assortative Mixing.

Esempio: la rete di amicizia di una scuola

Una scuola americana effettua un'analisi della coesione fra gli studenti basandosi sulla loro razza. Ecco il risultato:

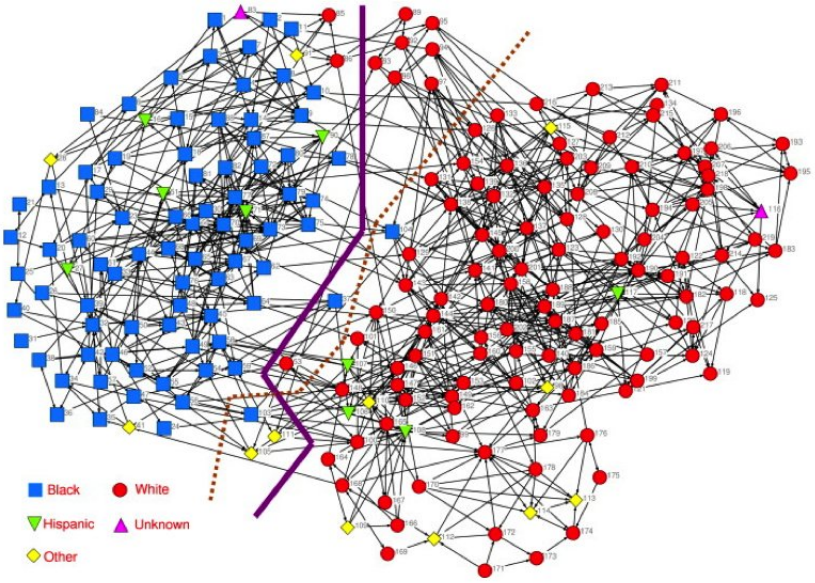


Figura 3.8: Vediamo due grossi gruppi omofili ben distinti.

3.7.1 Assortative Mixing discreto

Consideriamo i grafi i cui vertici vengono classificati in base ad alcune caratteristiche. Tali caratteristiche possono avere un numero finito di valori (non continuo). Possiamo pensare ad esempio a nazionalità, genere, lingua della pagina web, zona geografica, ecc.
La rete è detta *assortative* se una porzione significativa degli archi è stabilita fra vertici dello stesso tipo. Intuitivamente potremmo contare il numero di vertici scambiati fra nodi omofili e farne il rapporto col numero totale di vertici ma questa misura sarebbe uguale ad 1 quando tutti i veritci appartengono alla stessa classe e non ci piace. Vorremmo infatti cercare di evitare correlazioni banali e quindi poco significative.

Adottiamo una tecnica già vista: calcoliamo la frazione di archi tra vertici dello stesso tipo e da questo valore sottraiamo quegli archi che comunque ci aspetteremmo di trovare nel caso gli archi fossero stabiliti a caso. Abbiamo risolto il caso banale di prima, infatti con nodi della stessa classe il grafo randomico ci darebbe 1 e quindi la sottrazione sarebbe 0.

Calcoliamo gli archi tra vertici dello stesso tipo:

$$\sum_{edges(i,j)} = \delta(c_i, c_j) = \frac{1}{2}\sum_{ij} A_{ij}\delta(c_i, c_j) \tag{3.17}$$

Con $\delta(c_i, c_j)$ intendiamo il delta di Kronecker che vale 1 quando due elementi sono uguali e 0 quando due elementi sono diversi. In questo caso ci serve per capire se esiste un arco fra i due nodi.
Dopodiché calcoliamo gli archi attesi tra i vertici dello stesso tipo. Consideriamo un nodo i collegato ad altri k_i nodi. Vi sono in totale $2m$ possibili terminazioni nell'intera rete (poiché vi sono in totale m archi ed ogni arco ha 2 estremi). La possibilità che una delle terminazioni del nostro arco particolare sia una delle k_j che terminano in j è quindi $\frac{k_j}{2m}$.
Da ciò, il numero totale di archi attesi fra i e j è $\frac{k_i k_j}{2m}$.
Considerando tutti gli archi attesi abbiamo invece:

$$\frac{1}{2}\sum_{ij} \frac{k_i k_j}{2m}\delta(c_i, c_j) \tag{3.18}$$

È stato introdotto un 1/2 per evitare che si contino due volte le coppie di vertici.

Non resta che calcolare la differenza fra 3.17 e 3.18:

$$\frac{1}{2}\sum_{ij} A_{ij}\delta(c_i, c_j) - \frac{1}{2}\sum_{ij} \frac{k_i k_j}{2m}\delta(c_i, c_j) =$$

$$\frac{1}{2}\sum_{ij} \left(A_{ij} - \frac{k_i k_j}{2m}\right)\delta(c_i, c_j)$$

Oppure convenzionalmente usiamo la *modularità* Q che è lo stesso valore di sopra ma diviso per m.

$$Q = \frac{1}{2m}\sum_{ij} \left(A_{ij} - \frac{k_i k_j}{2m}\right)\delta(c_i, c_j) \tag{3.19}$$

Q restituisce valori molto minori di 1 solitamente. Nell'esempio in Figura 3.8 avevamo un valore di $Q = 0.305$ ed è assolutamente considerabile positive assortative mixing.

Normalizzazione della modularità

La modularità non è mai uguale ad 1 neanche in presenza di reti totalmente assortative. Normalizziamo quindi, come al solito, in base al valore massimo

che Q può raggiungere. Questo significa che dobbiamo avere che tutti gli archi del grafo sono stabiliti fra nodi dello stesso tipo ovvero $A_{ij} = 1$ qualora $\delta(c_i, c_j) = 1$. Quindi:

$$Q_{max} = \frac{1}{2m} \sum_{ij} \left(A_{ij} - \frac{k_i k_j}{2m} \right) \delta(c_i, c_j) =$$

$$Q_{max} = \frac{1}{2m} \left(2m - \frac{k_i k_j}{2m} \right) \delta(c_i, c_j) \tag{3.20}$$

Infine, normalizziamo come ormai siamo abituati facendo Q/Q_{max}.

$$r = \frac{Q}{Q_{max}} = \frac{\sum_{ij} \left(A_{ij} - \frac{k_i k_j}{2m} \right) \delta(c_i, c_j)}{\left(2m - \frac{k_i k_j}{2m} \delta(c_i, c_j) \right)} \tag{3.21}$$

Questo coefficente è noto come *assortativity coefficent* ma non è molto adoperato.

3.7.2 Assortative Mixing scalare

Non è sempre facile identificare classi precise con cui classificare un nodo. Pensiamo ad esempio alle stelle in un sistema di reccomandation: un voto basso può significare sia una che due stelle, così come un voto alto può significare quattro o cinque. Allo stesso modo possiamo pensare all'età, può essere utile non solo collegare nodi che hanno la stessa identica età bensì nodi con età *simili*. Non possiamo quindi più usare il delta di Kronecker. Possiamo quindi sfruttare alternativamente la *covarianza su tutti i vertici* di un arco.
Sia x_i il valore del vertice i della quantità scalare cui stiamo interessati. Consideriamo la coppia di valori (x_i, y_i) per i vertici agli estremi dell'arco (i, j) e calcoliamo la sua covarianza su tutti i vertici:

$$cov(x_i, x_j) = \frac{1}{2m} \sum_{ij} \left(A_{ij} - \frac{k_i k_j}{2m} \right) x_i x_j \tag{3.22}$$

Vediamo da dove arriva questa formula.
Iniziamo con contare la media del valore di x_i all'estremità di un arco:

$$\mu = \frac{\sum_{ij} A_{ij} x_i}{\sum_{ij} A_{ij}} = \frac{\sum_i k_i x_i}{\sum_i k_i} = \frac{1}{2m} \sum_i k_i x_i$$

Dopodiché sappiamo da dall'equazione 3.14 come calcolare la covarianza. Non ci resta che sostituire la media.

$$cov(x_i, x_j) = \frac{\sum_{ij}(x_i - \mu)(x_j - \mu)}{\sum_{ij} A_{ij}} =$$
$$\frac{1}{2m}\sum_{ij} A_{ij}(x_i x_j - \mu x_j - \mu x_i + \mu^2) =$$
$$\frac{1}{2m}\sum_{ij} A_{ij} x_i x_j + \mu^2 =$$
$$\frac{1}{2m}\sum_{ij} A_{ij} x_i x_j - \frac{1}{(2m)^2}\sum_{ij} k_i k_j x_i x_j =$$
$$\frac{1}{2m}\sum_{ij}\left(A_{ij} - \frac{k_i k_j}{2m}\right) x_i x_j$$

Ottenendo così la formula introdotta poco fa: è molto simile alla modularità Q.
Nel caso di covarianza positiva si avrà un bilanciamento reciproco fra i valori x_i e x_j quindi abbiamo *assortative mixing*. Nel caso contrario invece avremo valori negativi di covarianza e quindi *disassortative mixing*.

Normalizzazione della covarianza

Anche qui possiamo normalizzare la covarianza e ottenere un r così definito:

$$r = \frac{\sum_{ij}\left(A_{ij} - \frac{k_i k_j}{2m}\right) x_i x_j}{\sum_{ij}\left(k_i \delta_{ij} - \frac{k_i k_j}{2m}\right) x_i x_j}$$

3.7.3 Assortative mixing per grado

Si tratta di un caso particolare di assortative mixing scalare, il valore x_i è ora considerato il grado. Cioè nodi di grado simile si collegano facilmente? Oppure sono i nodi di grado differente a collegarsi più facilmente? Scriviamo la formula della covarianza usando come valore x_i il grado k_i.

$$cov(k_i, k_j) = \frac{1}{2m}\sum_{ij}\left(A_{ij} - \frac{k_i k_j}{2m}\right) k_i k_j$$

Come sempre, possiamo normalizzare:

$$r = \frac{\sum_{ij}\left(A_{ij} - \frac{k_i k_j}{2m}\right) k_i k_j}{\sum_{ij}\left(k_i \delta_{ij} - \frac{k_i k_j}{2m}\right) k_i k_j}$$

3.7.4 Influence

Anche il concetto di *influence* è molto importante ed è legato ma diverso dall'omofilia.

- L'omofilia è la tendenza degli individui ad aggregarsi poiché simili.
- L'influenza è la tendenza di invividui amici che, data la loro amicizia, diventano simili.

Il concetto di causalità è inverso nei due concetti!

Esempi di assortative mixing per grado

Rete	Tipo	N.Nodi	Assortativity
Co-autori in fisica	undirected	52 909	0.363
Co-autori in biologia	undirected	1 520 251	0.127
Co-autori in matematica	undirected	253 339	0.120
Collaborazioni nei film	undirected	449 913	0.208
Direttori di compagnie	undirected	7 673	0.276
Relazioni fra studenti	undirected	513	-0.290
Rete elettrica	undirected	4941	-0.003
Internet	undirected	10 697	-0.189
WWW	directed	269 504	-0.067
Interazione fra le proteine	undirected	2 115	-0.156
Rete metabolica	undirected	765	-0.240
Rete neurale	directed	307	0.226

Sono riportate tre categorie di reti: sociali, tecnologiche e biologiche. Ogni tipologia si comporta in maniera simile in termini di assortativity, tranne un caso nella rete delle relazioni fra gli studenti (ma è pur vero che il campione è di pochi individui quindi potrebbe essere non sufficiente per avere una misura sufficientemente esatta). Avere pochi campioni è un esempio di *bais* naturale, inclinazione, che indica che c'era un inclinazione iniziale che porta a risultati inconcludenti ed inaspettati.
Notiamo che tendenzialmente le reti sociali hanno assortativity maggiore di zero (probabilmente perché nelle società accade spesso che si creino dei sottogruppi molto connessi e quindi con nodi di grado simile fra loro).
Al contrario reti tecnologiche e biologiche hanno assortativity negative (probabilmente a causa del fatto che il grafico sia più semplice).

In generale possiamo pensare che reti più semplici sono naturalmente disassortative, mentre reti sociali grandi e complesse sono più naturalmente assortative.

Diminuire la complessità

Abbiamo detto che si usa il coefficiente di correlazione indicato dall'equazione 3.7.3. Ma tale formula ha complessità $O(n^2)$. Possiamo quindi usare una formula alternativa (complessità lineare):

$$r = \frac{S_1 S_e - S_2^2}{S_1 S_3 - S_2^2}$$

Dove:

$$S_e = 2 \sum_{edges(i,j)} k_i k_j \qquad S_2 = \sum_i k_i^2$$

$$S_1 = \sum_i k_i \qquad S_3 = \sum_i k_i^3$$

La somma in S_e ha m termini, mentre le somme S_i hanno n termini ciascuna dunque la complessità risulta essere $O(m+n)$.
La tecnica risulta ultile con un m nell'ordine di n (reti sparse), ma quando m è nell'ordine di n^2 (reti dense) questa formula non è così conveniente.

Capitolo 4

Memorizzare le reti

È importante studiare le strutture dati e gli algoritmi per la gestione delle reti e non limitarsi all'uso dei tools in maniera ignorante. Questo perché sapendo cosa c'è sotto possiamo capire al meglio i risultati restituiti dal tool, senza considerare che l'argomento è di frontiera e quindi può essere necessario sviluppare varianti di algoritmi o addirittura crearne di nuovi.
Gli algoritmi verranno trattati in seguito poiché richiedono nozioni che introdurremo a breve.

4.1 Strutture dati

I grafi vengono salvati in memoria tramite una struttura dati. Nonostante la matrice sia un buon modo per visualizzare una rete risulta un cattivo modo per memorizzare le informazioni del grafo poiché potremmo avere grafi molto sparsi dunque sprecare molto spazio. Non si dimentichi che è spesso necessario considerare una rete in senso dinamico e quindi memorizzare differenti stati di una stessa rete: ancora una volta la matrice risulta scomoda ed estremamente costosa. Formalizziamo questa idea.

4.1.1 Matrici

Una *matrice* risulta vantaggiosa in termini di accesso al nodo (abbiamo gli indici, $O(1)$) e di creazione di un nuovo arco (ancora $O(1)$). Nel caso poi di grafi non diretti possiamo avere una matrice triangolare e quindi usare solo metà matrice.
Gli svantaggi sono però nella lettura di tutti i nodi collegati ad un nodo (dobbiamo leggere l'intera riga anche se abbiamo un solo nodo vicino, quindi $O(n)$), lo spreco di spazio per la memorizzazione di grafi sparsi (molti 0) e anche la pesantezza in memoria: avendo 10GB di RAM e avendo come peso dei nodi un intero (4 Byte) possiamo rappresentare solamente 50.000 nodi.

4.1.2 Liste di adiacenza

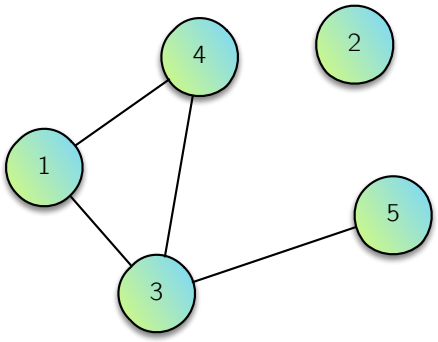
Decisamente più sensate sono le *liste di adiacenza*: abbiamo una lista di liste. Ogni sotto-lista conterrà i nodi adiacenti al nodo indiciato sulla lista principale. Dovremo salvare al massimo $2m$ interi e si ricorda che nel caso di grafi sparsi $2m$ è decisamente minore di n^2. Le informazioni mantenute sono solamente quelle utili, quindi il tempo di accesso per le operazioni è decisamente ridotto, nella fattispecie:

- Inserimento di un arco: $O(1)$.
- Ricerca / eliminazione di un arco: $O(m/n)$ al caso medio, $O(1)$ per grafi sparsi (poiché $m \sim n$).
- Eliminazione di tutti i vicini di un nodo n: $O(m/n)$ al caso medio, $O(1)$ per grafi sparsi.

Tendenzialmente le liste di adiacenza sono la struttura più utilizzata per memorizzare reti.

Esempio di lista di adiacenza

Vediamo un esempio di lista di adiacenza.



Nodo	Vicini
1	3, 4
2	
3	1, 4
4	1, 3, 5
5	4

4.1.3 Alberi di adiacenza

Sono simili alle liste di adiacenza ma ogni riga è memorizzata come un albero (bilanciato e binario, per le massime performances) invece che un semplice array. Raggiungiamo le seguenti complessità:

- Inserimento di un arco: $O(\log_2 k)$ dove k è il numero di vicini.
- Ricerca: $O(k)$.
- Eliminazione di un arco: $O(\log_2 k)$.
- Eliminazione dei vicini di un vertice: $O(k)$ (come già per le liste).

4.1.4 Heap e ulteriori strutture

Gli *heap* sono alberi con indice. Sono utili per mantenere gli attributi dei nodi e non per la rete in sé.

Troviamo inoltre in letteratura altre strutture:

- Soluzioni ibride matrice/lista.
- Pesi e attributi sugli archi sono salvati in modo semplice (hash table).
- Edge list (lista degli archi), poco efficiente ma molto intuitivo.

Capitolo 5

Struttura di grandi reti

5.1 Introduzione

Abbiamo introdotto strumenti matematici per formalizzare reti, misure e metriche per analizzarne quantitativamente la struttura e abbiamo anche discusso di come strumenti di visualizzazione possano aiutare l'analisi di una rete e in particolare quali aspetti siano in grado di evidenziare.
Nella rete possiamo però trovare pattern ricorrenti che è interessante *cercare*, poiché ci danno informazioni importanti ed essenziali. Obiettivo di questo capitolo è quello di capire questi pattern e di darne una definizione precisa e formale. Osserveremo fenomeni quali:

- Componenti di dimensioni variamente distribuite dette *largest connectet component*(possono poi essere *strong* o *weak*).
- Distanze medie molto piccole (*small word effect*).
- Distribuzione di grado che presenta *asimmetria statistica*.
- *Leggi di potenza* e *invarianza di scala*.
- Valori del *coefficiente di correlazione del grado* in grado di 'classificare' la tipologia delle reti (positive, negative, neutre).

5.2 Componenti

Nelle reti indirette c'è, in genere, almeno una componente notevolmente più grande delle altre. Tale componente è infatti tendenzialmente maggiore del 50% dei nodi della rete ed alcune volte può essere anche > del 90%. Definiamo quella componente come *componente connessa più grande* (LCC, Largest Connected Component).
Si consideri che nella rete degli attori solamente il 2% non fa parte della LCC.

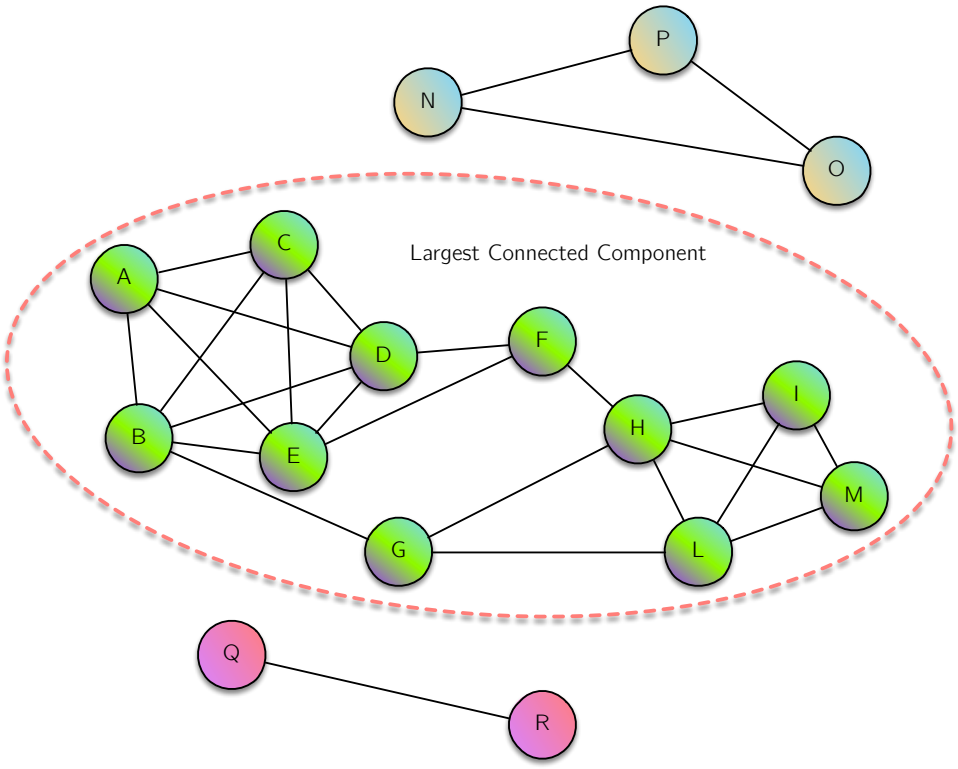


Figura 5.1: Esempio di LCC.

Nel caso in cui tutti i nodi della rete facciano parte dell'LCC siamo in un caso particolare, probabilmente per costruzione non è stato possibile raggiungere i nodi non connessi (rete dei router internet) oppure c'è stato un bias nella raccolta dei dati.
All'interno di una rete troveremo sempre una sola LCC (tranne rarissimi casi). Lo possiamo capire in termini di probabilità: supponiamo una rete di n nodi. Diciamo dunque di avere due LCC, ognuna con $\frac{n}{2}$ nodi. Ogni LCC potrà avere un numero di archi circa uguale a $\frac{n^2}{2} = \frac{n^2}{4}$, ed il numero di archi totali del grafo sarà circa uguale a $\frac{n^2}{2}$. Paragonando queste due quantità capiamo che per ottenere due LCC i singoli archi di ogni LCC dovrebbero essere guidati nell'evitare in tutto e per tutto l'altra LCC e questo è estremamente improbabile che accada naturalmente.

Nelle reti dirette

Nel caso di reti dirette distinguiamo fra *Strongly CC* (*SCC*) e fra *Weakly CC* (*WCC*). Se una componente è SCC per ogni coppia di nodi nella componente

esiste un cammino di andata ed uno di ritorno, mentre se una componente è WCC per ogni coppia di nodi nella componente esiste almeno un cammino.

5.3 Cammini, distanze e diametro

Cosa sappiamo dei cammini di una rete? Milgram, con il suo esperimento, ha suggerito il fatto che i cammini minimi di una rete hanno *lunghezza media molto bassa*, ovvero un grafo casuale con n nodi abbiamo che i cammini crescono nell'ordine di $\log(n)$.
Si ricorda che nel calcolo dei cammini possiamo incappare in valori infiniti quando due nodi non sono collegati (appartengono a componenti differenti) (per approfondimenti si veda la Sezione 3.1.2).

5.3.1 Il diametro

Un altro interessante parametro da considerare è il diametro: il *valore massimo tra tutte le distanze di lunghezza finita*.
Anche il diametro cresce seguendo un andamento logaritmico e talvolta ci sono reti che sviluppano un *core* ove il diametro cresce nell'ordine di $\log(\log(n))$: si vede molto bene questo ultimo fenomeno nell reti *scale-free* (dove c'è invarianza di scala, ne parleremo meglio in seguito). È una misura difficile da verificare precisamente con un approccio empirico (pochi dati).

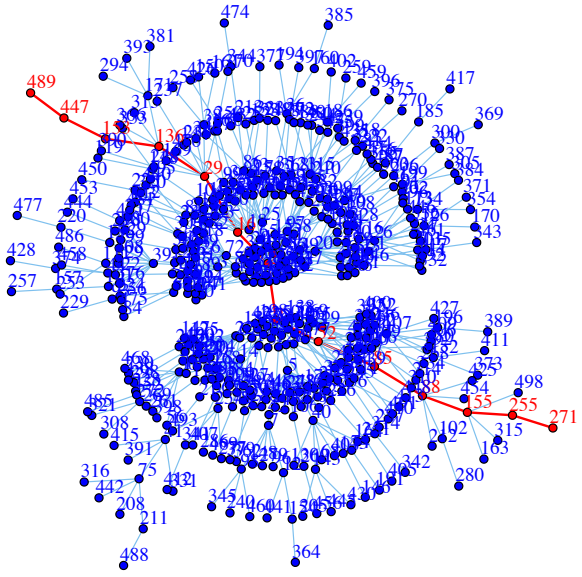


Figura 5.2: Diametro in un grafo ad invarianza di scala.

Nel grafico sopra il diametro (in rosso) vediamo che è decisamente fuori media,

con lunghezza nell'ordine di $\log(n)$. Inoltre notiamo la disposizione dei nodi concentrica per via dell'invarianza di scala: al centro c'è un core, un gruppo più massiccio di nodi, ove la media cresce $\log(\log(n))$.

5.3.2 Effetto funneling

Tale effetto si vede bene nelle reti sociali (già Milgram l'aveva delineato): tra i vari vicini di una rete sociali sociale solamente pochi sono molto connessi e responsabili della trasmissione di informazioni. Questo fenomeno è comprensibile, si pensi ai gruppi sociali che una persona frequenta: quelli della palestra, i compagni studio, gli amici della fidanzata, ecc.
Attenzione, questo effetto non è sempre presente, ad esempio possiamo pensare che non ci sia nella rete Internet.

5.4 Distribuzione del grado

Fra le proprietà più interessanti troviamo la distribuzione del grado. Il grado è quantità di archi entranti/uscenti o totali da un nodo. Il modo più intuitivo di trattare la distribuzione di grado di un grafo è visualizzarlo tramite la sua frequenza. Prendiamo questo grafo:

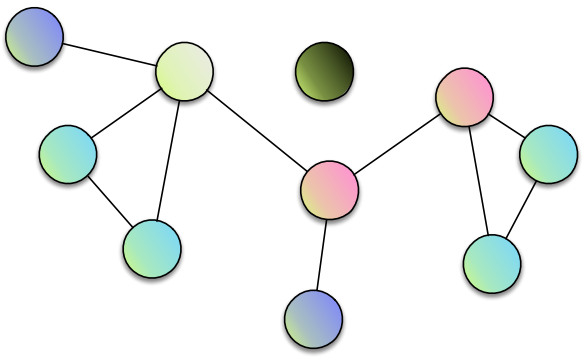


Figura 5.3: Nodi con colori uguali hanno frequenza uguale.

Definiamo con p_k la quantità frequenza di nodi con grado k all'interno del grafo. Nell'esempio le fequenze sono:

k	frequenza
p_0	1/10
p_1	2/10
p_2	4/10
p_3	2/10
p_4	1/10

È spesso considerata la *distribuzione comulativa*, somma cumulativa delle singole distribuzioni:

$$P_k = \sum_{k' \neq k}^{\infty} p'_k \tag{5.1}$$

Chiaramente nelle reti reali, ci fermeremo a un certo k_{max}.

5.4.1 Introduzione: distribuzioni frequenti e asimmetria

Possiamo trovare delle distribuzioni simili per reti con strutture diverse, le più frequenti in pratica sono la distribuzione *esponenziale* e la distribuzione a *legge di potenza*.
In queste distribuzioni si osservano:

- Fenomeni di *asimmetria*,
- Presenza di una coda *pesante* da una delle due parti della distribuzione,
- Pochi hub (pochi nodi molto connessi),
- Legge dell'80-20 (*Pareto*).

La legge dell'80-20 deriva da un esperimento effettuato da Pareto nei primi anni del '900. Tale esperimento delinava la distribuzione di redditi di una popolazione, facendo notare che pochi membri avessero la altissimo reddito (20Usiamo istrogrammi per rappresentare i nostri dati, così da visualizzarli in modo più leggibile. Proprio grazie a questa rappresentazione è facile vedere l'asimmetria di cui parlavamo prima: Proprio per la loro costituzione è interes-

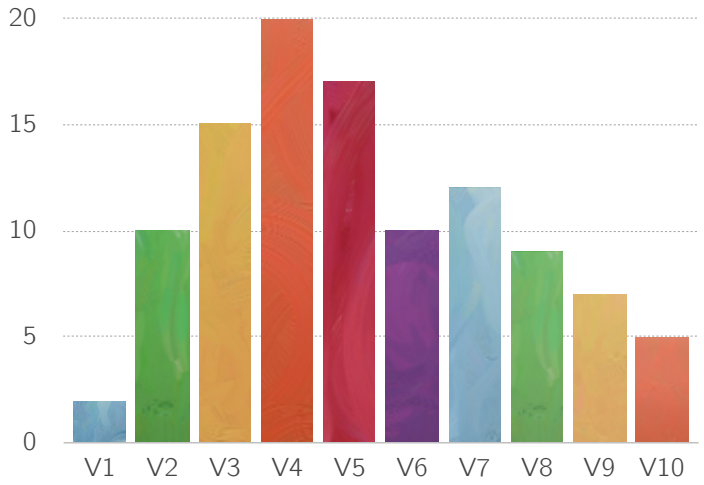


Figura 5.4: Esempio di istrogramma asimmetrico

sante notare che la *media* di una distribuzione a legge di potenza o esponenziale è assolutamente non interessante e ci dice poco.

Sui plot

Le distribuzioni di cui abbiamo sopra sono molto sbilanciate in termini di frequenza dei valori sui due assi. Dunque non si usa mai una rappresentazione *lin-lin* ma si predilige una rappresentazione *lin-log* per le scale esponenziali e *log-log* per la legge di potenza. Vediamo due esempi rispettivamente di distribuzione esponenziale e di distribuzione a legge di potenza :

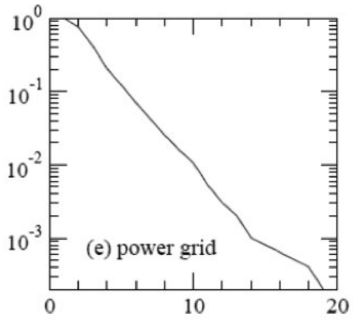


Figura 5.5: $P_k \propto e^{-\alpha k}$

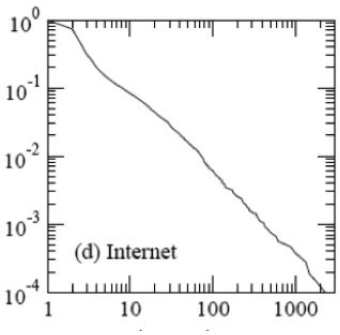


Figura 5.6: $P_k \propto k^{-\alpha}$

Vediamo la visualizzazione per esempio di una PDF (probability distribution function) con la sua CDF (cumulative distribution function):

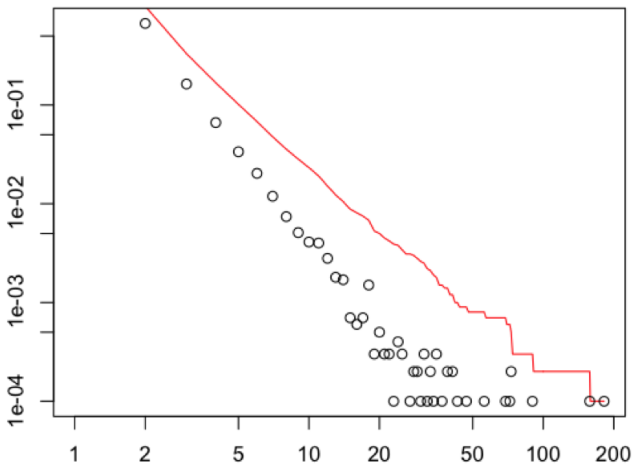


Figura 5.7: Distribuzione cumulativa (in rosso).

5.4.2 Leggi di potenza

La legge di potenza segue questa espressione: $k^{-\alpha}$. Guardando un plot della distribuzione notiamo istintivamente una *linea retta* che approssima i punti: $\ln p_k = -\alpha \ln k + c$ dove α e c sono costanti e $\alpha > 0$ è la *pendenza*.

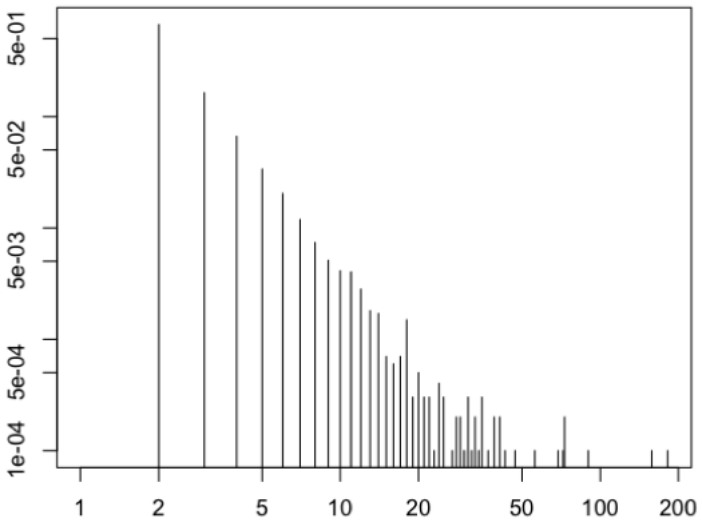


Figura 5.8: Grafico di una power-law.

Facendo l'esponenziazione ad entrambi i membri troviamo l'espressione di cui parlavamo prima:

$$p_k = Ck^{-\alpha} \tag{5.2}$$

Riguardando la figura notiamo però che in coda la legge non vale più. Questo perché nella realtà la legge non è seguita in tutto il range dei dati a disposizione ma in testa o in coda la distribuzione può variare. Per risolvere questo problema in genera si applica un *cut-off* dei dati (con la conseguenza di una minore rappresentazione statistica).
Ora che abbiamo capito la loro costituzione, è importante capire perché le studiamo: le leggi di potenza sono molto frequenti in natura, nel nostro caso le troviamo nella distribuzione di grado. Le reti che seguono una power-law sono dette *reti ad invarianza di scala*. Si noti che la costante C non è importante a differenza della pendenza α che è molto rilevante. Tale pendenza è spesso compresa fra 2 e 3.

Zipf, Pareto e legge di potenza

Siamo stati imprecisi. In realtà le leggi di potenza sono indicabili in tre modalità diverse: in generale una legge di potenza descrive fenomeno per i quali eventi con valori alti sono rari mentre sono molto frequenti eventi con valori bassi.

Classici esempi sono le grandezze delle città, le magnitudo dei terremoti e la frequenza delle parole. Andiamo dunque a distinguere queste tre differenti modalità per definire le leggi di potenza:

La legge di Zipf. Zipfs era un professore di linguistica ad Harvard. I suoi studi si sono orientati sulla frequenza delle parole nei testi inglesi e ha trovato che la funzione che restituiva la frequenza y della $r-$esima parola più frequente seguiva la legge:

$$y \approx r^{-b} \tag{5.3}$$

con b molto vicina ad 1.

La legge di Pareto. Pareto è stato un ingegnere, storico e sociologo franco-italiano. È famoso per il suo studio in merito alla *sperequazione della ricchezza*. La sua concentrazione non si è posta sull'r-esimo reddito più grande bensì su quante persone avevano un reddito maggiore di x: in sostanza si ha una versione cumulativa di quanto scoperto da Zipf:

$$P[X > x] \approx x^{-b} \tag{5.4}$$

Legge di potenza. Una legge di potenza non è altro che una PDF associata alla CDF: otteniamo, nell'esempio di Pareto, il numero di persone che hanno esattamente un reddito pari a x.

$$P[X = x] \approx x^{-(b+1)} = x^{-\alpha} \tag{5.5}$$

dove l'esponente della power-law è $\alpha = b + 1$ con b l'esponente della distribuzione di Pareto.

Invarianza di scala

Spesso la legge di potenza è detta *invarianza di scala*, cioè una distribuzione che rimane uguale a se stessa indipendentemente dalla scala in cui la osserviamo. In formule possiamo rappresentare l'invarianza di scala come:

$$p(bx) = g(b)p(x) \tag{5.6}$$

Cosa significa questa formula? Prendiamo due valori x_1 e x_2. Abbiamo dunque che:

$$p(bx_1) = g(b)p(x_1) \Rightarrow g(b) = \frac{f(bx_1)}{p(x_1)}$$

$$p(bx_2) = g(b)p(x_2) \Rightarrow g(b) = \frac{f(bx_2)}{p(x_2)}$$

e quindi

$$\frac{f(bx_1)}{p(x_1)} = \frac{f(bx_2)}{p(x_2)}$$

Come vediamo, al variare della scala (b), l'equivalenza è mantenuta.

Esistono altre distribuzioni scale-free?

Abbiamo detto che una distribuzione power-law è sempre ad invarianza di scala. Esistono altre distribuzioni che godono di questa interessante proprietà?
Prendiamo la definizione di invarianza di scala (Formula 5.6) e applichiamola ad $x = 1$.

$$p(b) = g(b)p(1) \Rightarrow g(b) = \frac{p(b)}{p(1)}$$

Dopodiché prendiamo quanto appena trovato e la re-inseriamo nella definizione sostituendo $g(b)$:

$$p(bx) = p(x) \cdot \frac{p(b)}{p(1)} = \frac{p(b)p(x)}{p(1)}$$

Dato che tale equazione è vera $\forall\ b$ possiamo derivare entrambi i lati rispetto a b:

$$xp'(bx) = \frac{p'(b)p(x)}{p(1)}$$

che con $b = 1$ risulta:

$$x\frac{dp}{dx} = \frac{p'(1)}{p(1)}p(x)$$

Risolviamo questa equazione differenziale di primo grado e otteniamo:

$$\ln(p(x)) = \frac{p'(1)}{p(1)}\ln(x) + c$$

Abbiamo ritrovato l'equazione della retta che avevamo notato nel plot! Procediamo facendo l'esponenziazione:

$$p(x) = Cx^{\frac{p'(1)}{p(1)}}$$

E abbiamo quindi ottenuto che con $\alpha = -\frac{p(1)}{p'(1)}$:

$$p(x) = Cx^{-\alpha}$$

Che è la nostra definizione di power-law.
Ne deduciamo che abbiamo invarianza di scala avremo necessariamente una legge di potenza e viceversa.

5.4.3 Determinare una power-law empiricamente

L'approccio più semplice per determinare una power-law è quello di prendere la distribuzione del grado e metterla su un grafico log-log: se siamo in grado di tracciare una linea fra tutti i punti del grafico siamo quasi certi che si tratti

di una power-law.
Vi sono alcuni problemi con questo approccio: come abbiamo già detto è possibile che in coda non sia possibile tracciare una linea retta e putroppo quella zona è quella che ci interessa di più, inoltre quando abbiamo molti dati tenderemo ad avere degli intervalli ampi rappresentati sulle ascisse e quindi avremo una crescita esponenziale da intervallo ad intervallo.
Le soluzioni sono due:

- Si cambia modalità di *visualizzazione* dei singoli intervalli (detti *bin*) dell'istogramma.
- Si usano intervalli *più ampi* (col rischio di perdita di informazioni).

Ciò che si usa nella realtà è un compromesso, si usano *bin* più ampi alla fine e più stretti all'inizio. La tecnica che permette il dimensionamento diverso di intervalli sull'istogramma è detta *logarithmic binning*.

Logarithmic binning

Lo schema allarga ogni intervallo di un certo fattore a. Abbiamo quindi:

- bin 1: $1 \leq k < 2$ (spessore $a = 2$)
- bin 2: $2 \leq k < 4$ (spessore $a^2 = 4$)
- bin 3: $4 \leq k < 20$ (spessore $a^3 = 16$)
- ...
- bin n: $a^{n-1} \leq k < a^n$ (spessore $a^n - a^{n-1} = (a-1)a^{n-1}$)

Notiamo che in scala logaritmica gli intervalli sembreranno uguali e notiamo anche che non sono rappresentati i nodi non connessi ($k = 0$) che dovranno essere esaminati a parte.

Osserviamo ora una power-law (Figura 5.9) e riflettiamoci sopra:

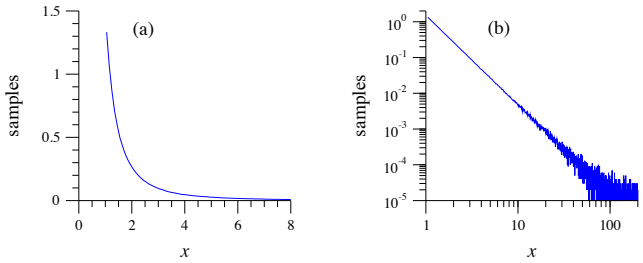


Figura 5.9: Osseriviamo una power-law.

Partiamo, in alto a sinistra, da un'analisi del degree visualizzata in scala lineare. Per vederci chiaro convertiamo la scala in log-log (in alto a destra) abbiamo il caso 'perfetto'! La retta è evidente, purtroppo però abbiamo un agglomerato di dati sulla coda. Adottiamo quindi il logarithmic binning (Figura 5.10): ora è tutto molto più leggibile.

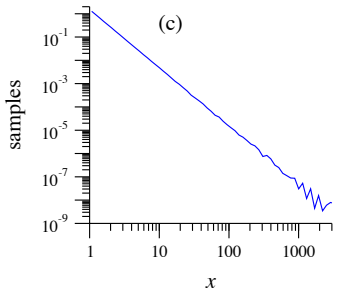


Figura 5.10: Applicazione del logarithmic binning.

Usare la CDF

Un approccio alternativo è l'uso della CDF (Formula 5.1). Siamo interessati a capire se in coda (cioè per un $k < k_{min}$) p_k segue una legge di potenza o meno. Combinando la CDF e la definizione di legge di potenza (Formula 5.2) otteniamo:

$$p_k = Ck^{-\alpha} \Rightarrow P_k = C \sum_{\bar{k}=k}^{\infty} \bar{k}^{-\alpha}$$

Considerando però che la power-law è una funzione che *cresce molto lentamente* al variare di k possiamo usare un integrale al posto di una sommatoria.

$$p_k = Ck^{-\alpha} \approx P_k = C \int_k^{\infty} \bar{k}^{-\alpha} d\bar{k} = \frac{C}{\alpha - 1} k^{-(\alpha-1)}$$

Tutto ciò è vero solamente per α maggiore di uno, altrimenti l'integrale non converge.
La somiglianza che avevamo notato in termini di pendenza quando abbiamo introdotto la CDF è ora più chiaro. Infatti abbiamo che:

$$p_k = Ck^{-\alpha} \Rightarrow P_k = C'k^{-(\alpha-1)}$$

Dove $C' = \frac{C}{1-\alpha}$. C'è solo una differenza di 1 nell'esponente di k.

Riassumendo, per verificare se la nostra distribuzione di grado è power-law calcoliamo la CDF, disegniamo un plot log-log e cerchiamo una linea dritta.

Vantaggi della CDF. Usare la CDF è vantaggioso rispetto a usare i dati della frequenza poiché:

- La CDF è facile da calcolare, possiamo usare librerie dedicate già presenti.
- Non è necessario effettuare *binning*.
- Preserva tutte le informazioni.

Svantaggi della CDF. Putroppo vi sono anche i seguenti svantaggi:

- È più difficile da interpretare (se guardiamo il valore in un k preciso abbiamo la sommatoria delle frequenze dei valori di k da 0 fino a quel k e non il k preciso che volevamo esaminare).
- Punti successivi nel plot sono correlati dunque non c'è indipendenza. Questo significa che non è possibile estrarre l'esponente della power-law dalla CDF poiché tendenzialmente il calcolo è usato adoperando metodi *least squares* che assumono l'indipendenza fra punti.

Da ciò deduciamo che la CDF è molto utile per farci capire 'ad occhio' se abbiamo in mano una distribuzione power-law o meno. Dopodiché per ottenere qualche dato effettivamente utilizzabile dovremo rieffettuare il calcolo tramite PDF.

5.4.4 Ulteriori misure

Calcolo della pendenza α

La pendenza della nostra retta è α. Dato k_{min} il valore minimo per cui la legge di potenza è verificata e dato N il numero di vertici tali che $k \leq k_{min}$ (cioè che hanno grado superiore al minimo) eseguiamo una sommatoria (solamente per i nodi che soddisfano questa condizione):

$$\alpha = 1 + N \left[\sum_i \ln \frac{k_i}{k_{min} - \frac{1}{2}} \right]^{-1} \tag{5.7}$$

Ci sta bene imporre un k_{min}, spesso il calcolo per i nodi molto piccoli di una power-law non è importante.

Calcolo dell'errore statistico su α

Prendiamo per esempio il calcolo dei dati su internet: abbiamo un errore di $\alpha = 2.11 \pm 0.01$. L'errore statistico σ si calcola come segue:

$$\sigma = \sqrt{N} \left[\sum_i \ln \frac{k_i}{k_{min} - \frac{1}{2}} \right]^{-1} = \frac{\alpha - 1}{\sqrt{N}}$$

5.4.5 Proprietà delle power-law

Passiamo ora in rassegna alcune interessanti proprietà relative alle power-law.

Normalizzazione della power-law

La costante C non piace così com'è perché ci piacerebbe fosse definita in modo tale da normalizzare tutto P_k, ovvero vorremmo che la somma dei p_k fosse uguale a 1. In formule:

$$\sum_{k=1}^{\infty} p_k = 1 \Rightarrow C \sum_{k=1}^{\infty} k^{-\alpha} = 1$$

E quindi:

$$C \frac{1}{\sum_{k=1}^{\infty} k^{-\alpha}} = \frac{1}{\zeta(\alpha)}$$

Abbiamo introdotto ζ, la *zeta di Reinmann*. La power-law normalizzata risulta quindi essere:

$$p_k = \frac{k^{-\alpha}}{\zeta(\alpha)} \tag{5.8}$$

con $k > 0$ e $p_0 = 0$.

La normalizzazione risulta utile per aspetti formali ma è poco utilizzabile in ambito empirico , in realtà si usa la funzione ζ incompleta dove k parte da k_{min} e non da 1.:

$$p_k = \frac{k^{-\alpha}}{\zeta(\alpha), k_{min}}$$

In alternativa è possibile usare un integrale al posto della sommatoria.

Momenti di una distribuzione p_k

Il primo momento è la media:

$$\langle k \rangle = \sum_{k=0}^{\infty} k p_k$$

Il secondo momento è la media quadratica:

$$\langle k^2 \rangle = \sum_{k=0}^{\infty} k^2 p_k$$

E l'm-esimo momento:

$$\langle k^m \rangle = \sum_{k=0}^{\infty} k^m p_k$$

Supponiamo di avere una power law che si manifesta solo da k_{min} in poi. Spezziamo la sommatoria sopra e otteniamo:

$$\langle K^m \rangle = \sum_{k=0}^{k_{min}} k^m p_k + c \sum_{k=k_{min}}^{\infty} k^{m-\alpha}$$

Ancora una volta possiamo approssimare con l'integrale:

$$\langle K^m \rangle \approx \sum_{k=0}^{k_{min}} k^m p_k + c \int_{k=k_{min}}^{\infty} k^{m-\alpha} dk$$

La prima parte è finita, la seconda si risolve ottenendo:

$$\langle K^m \rangle \approx \sum_{k=0}^{k_{min}} k^m p_k + \frac{c}{m-\alpha+1} \left[k^{m-\alpha+1} \right]_{k_{min}}^{\infty}$$

Quando $m \geq \alpha + 1$ abbiamo che la seconda parte tende a infinito e quindi il tutto tende a infinito.
In particolare abbiamo che $\langle k^2 \rangle$ è finito solo se $\alpha > 3$ ma noi sappiamo che nei casi reali α è spesso compreso fra 2 e 3: ci dobbiamo quindi attendere un secondo momento infinito.
Questo significa che varianza, media e deviazione standard sono inutili.
È pur vero che nelle reti reali abbiamo valori finiti e quindi avremmo un k_max ma ad esempio in Internet il calcolo si effettua su un $n \approx 20000$: capiamo bene che nel nostro contesto è comunque un numero enorme, come fosse infinito.

Curve di Lorenz

Consideriamo la frazione degli archi connessi ai vertici con grado più alto (ci troviamo in una condizione di legge di potenza pura). Si può dimostrare che vale:

$$W = P^{\frac{\alpha-2}{\alpha-1}}$$

dove W è la frazione di archi connessi alla porzione P di vertici di grado più alto.
Possiamo quindi rappresentare le due misure su un grafico (Figura 5.11), in questo esempio P è la frazione di popolazione mentre W è il benessere.
Più basso è il valore di α e più le curve diventano concave. Per valori di poco superiore a 2 le curve incrementano molto velocemente ciò significa che una ampia porzione di archi sono connessi ad una piccola porzione di nodi: ecco da dove deriva la famosa legge dell'80-20 (precisamente 86-20). Un esempio sono le pagine web: il 50% delle pagine web possiedono l'89% dei collegamenti.

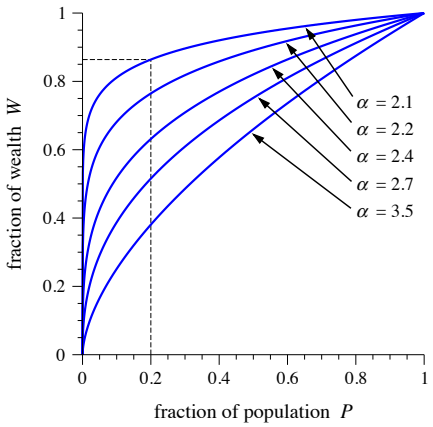


Figura 5.11: Le curve di Lorenz

5.5 Distribuzioni di altre misure

5.5.1 Misure di centralità

Abbiamo visto molte misure di centralità (betweenness, closeness, eigenvector) ed ognuna di esse ha la sua distribuzione... Nessuna di queste però è interessante come quella del grado (che abbiamo studiato nel dettaglio). Tendenzialmente si hanno distribuzioni asimmetriche (ma non sempre leggi di potenza).

5.5.2 Coefficiente di clustering

Il coefficiente di clustering misura la *densità di triangoli* in una rete (due miei amici sono amici fra loro). Il clustering viene misurato tramite il controfondo con una rete casuale R che possiede lo stesso numero di vertici e di archi della rete che stiamo studiando. Nel caso di grafo random definiamo C_r come segue:

$$C_R = \frac{1}{2}\frac{\left[\langle k^2\rangle - \langle k\rangle\right]^2}{\langle k\rangle^3} \tag{5.9}$$

dove k e k^2 assumono valori finiti.
Notiamo che al crescere di n diminuisce C_R ciò significa che nella rete non ci sono hub. Tendenzialmente nelle reti sociali $C \gg C_R$; in Internet invece $C \ll C_r$. Si ricorda inoltre lo studio dei motifs (Sezione 1.2.3) e del clustering sul singolo nodo (Sezione 1.2.3). In molti casi un pattern interessante che emerge è che vertici con grado alto tendono ad avere un clustering locale basso.

Capitolo 6

Modelli per reti

I modelli sono gli strumenti fondamentali grazie ai quali possiamo lavorare con le reti in maniera costruttiva e quindi sfruttare quanto abbiamo imparato fin ora in maniera rigida e formale.

6.1 Introduzione

6.1.1 Cosa tratteremo

Facciamo una breve lista dei modelli di grafi che tratteremo in seguito:

- Grafi random, modello Erdös-Rényi (statici)
- Grafi random con distribuzione arbitraria (statici)
- Modelli di grafi dinamici
- Modello small world (con attenzione al clustering e al rapporto fra clustering e diametro)
- Navigabilità di Kleinberg

Poi faremo alcuni cenni sulla teoria dei giochi.

6.1.2 Perché i modelli

I modelli ci sono utili poiché formalizzano una situazione altrimenti complessa da visualizzare. Ci possono essere utili per vedere l'evoluzione di una rete nel tempo (ad esempio le reti epidemiche), ecc. Fra i processi dinamici interessanti da studiare troviamo:

- Epidemie
- Consenso

- Colorazione (scegliere il colore di un nodo in modo che sia diverso da tutti gli adiacenti)
- Voting (scegliere il colore di un nodo in modo che sia diverso da tutti gli adiacenti)

6.2 I grafi random

6.2.1 Definizioni preliminari

Il modello di grafo $G(n, m)$ è costituito da n nodi ed m archi. Definito Ω la famiglia (insieme) di tutti i grafi composti da n nodi ed m archi, abbiamo che la probabilità che un grafo sia proprio un certo G è pari a:

$$P(G) = \frac{1}{\mid \Omega \mid}$$

Diametro medio

Possiamo definire il diametro medio fra tutti i grafi generabili come (ℓ è il diametro di un grafo):

$$\langle \ell \rangle = \sum_{\forall G \in \Omega} P(G) \cdot \ell(G) = \frac{1}{\mid \Omega \mid} \cdot \sum_{\forall G} \ell(G)$$

Grado medio

Possiamo anche definire il grado medio ($2m$ sono tutti gli archi generabili) di un singolo grafo (l'avevamo già definito così):

$$\langle k \rangle = \frac{2m}{n}$$

Definizioni in termini di P

In realtà risulta molto più interessante definire il grafo come $G(n, p)$ ovvero in termini di:

- n: numero di vertici del grafo
- p: probabilità che ogni coppia di vertici sia unita da un arco

In questo modo si rende decisamente più interessante provare a variare p e non direttamente m, così da capire quale m viene generato e da studiare il grafo in termini più generici. Data questa definizione un grafo G appare con probabilità:

$$P(G) = p^m (1-p)^{\binom{n}{2} - m} \tag{6.1}$$

La giant component

Una relazione importante è quella fra la giant component (LCC) e il numero di nodi. Più precisamente abbiamo che, esaminando

$$\frac{\text{giant component}}{n}$$

Se con $n \to \infty$ otteniamo una costante allora $n \propto$ giant component cioè la giant component permane e si allarga, se invece abbiamo che il risultato è 0 significa che avremmo un grafo totalmente sparso con nodi non collegati (e quindi nessuna giant component).

6.2.2 Distribuzione di grado

Abbiamo il nostro bel grafo definito come $G(n, p)$. Sappiamo che p è la probabilità che fra due nodi esista un arco, dunque la probabilità che non vi esista vale $1 - p$.
Possiamo perciò definire la probabilità che un nodo abbia un certo grado k come bernoulliana $B(n-1, k, p)$ dove $n-1$ sono i possibili nodi da scegliere (non consideriamo self-loops) e p è la probabilità di connessione. Sappiamo che:

$$B(n-1, k, P) = \binom{n-1}{k} p^k (1-p)^{n-1-k} \tag{6.2}$$

E sappiamo anche che le bernoulliane sono approssimabili a poissionane se p è molto piccolo oppure ad una normale se n è molto grande.

In ogni caso tutte e tre queste distribuzioni sono *a coda leggera*, ovvero decrescono velocemente... questo ci dice che non ci sono hub nella rete! Ma come abbiamo detto in precedenza la distribuzione di grado spesso ha una distribuzione power-law che è invece a *coda pesante*; iniziamo quindi ad intuire un difetto di questa rappresentazione troppo semplicistica che potrebbe, in certi casi, non rispettare la realtà.

6.2.3 Media degli archi

Come abbiamo detto il numero di archi m non è stabilito in precedenza ma è legato alla probabilità p, dunque è possibile stimarne un valore.
Il numero di grafi con esattamente n vertici e m archi è equivalente al numero di modi di piazzare gli archi fra le $\binom{n}{2}$ coppie di distinte di vertici. Ogni grafo ha probabilità di apparire $P(G)$ come definito nell' Eq. 6.1, pertanto la probabilità di avere un grafo con m archi è i modi di prendere quegli archi per la probabilità che quel grafo appaia:

$$P(m) = \binom{\binom{n}{2}}{m} p^m (1-p)^{\binom{n}{2}-m} \tag{6.3}$$

Tale equazione è la definizione della distribuzione binomiale, perciò il valore medio di m (su tutti i grafi) risulta essere:

$$\langle m \rangle = \sum_{m=0}^{\binom{n}{2}} mP(m) = \binom{n}{2} p \tag{6.4}$$

Il risultato non ci stupisce, stiamo dicendo che il numero atteso di archi fra qualsiasi coppia è la scelta della coppia per la probabilità che l'arco ci sia.

6.2.4 Grado medio

Possiamo usare il risultato appena ottenuto per calcolare il grado medio di un vertice. Sappiamo che in grafo con esattamente m archi i vertici hanno grado medio $\langle k \rangle = 2m/n$. Sostituendo questo valore ad m in Eq. 6.4 otteniamo:

$$\langle k \rangle = \sum_{m=0}^{\binom{n}{2}} \frac{2m}{n} P(m) = \frac{2}{n}\binom{n}{2} p = (n-1)p \tag{6.5}$$

In letteratura questo valore è definito come c pertanto scriveremo:

$$c = (n-1)p \tag{6.6}$$

Anche questa volta abbiamo un risultato che non ci sconvolge: diciamo che il numero atteso di archi connessi a un vertice è equivalente alla probabilità p che vi sia l'arco per la quantità degli altri archi.

6.2.5 Coefficiente di clustering

Il coefficiente di clustering ci dice quando 'due miei amici sono amici fra loro'. Sfruttando quanto appena trovato in Eq. 6.6 sappiamo che la probabilità che qualsiasi coppia di vertici abbia un arco fra loro è sempre uguale e vale $p = c/(n-1)$, pertanto il coefficiente di clustering risulta:

$$C = \frac{c}{n-1} \tag{6.7}$$

Anche qui notiamo una discrepanza con la realtà, dato c costante ed $n \to \infty$ l'Eq. 6.7 tende a 0, mentre sappiamo che nella realtà i coefficienti di clustering sono spesso molto alti anche in grafi con moltissimi nodi.

6.2.6 Andamento della giant component

Cerchiamo ora di capire come varia la giant component al variare di p. Possiamo subito dire che:

- Se $p = 0$ la giant component è costante e vale 1, cioè un singolo nodo.

- Se $p = 1$ la giant component è tutto il grafo (n nodi).

È interessante capire come avviene la transizione fra questi due stati: ad esempio, è graduale? No, in realtà la transizione è netta ed accade 'ad un certo punto'. Dimostriamolo seguendo la dimostrazione sviluppata da Erdös e Rényi.

Dimostrazione

Definiamo con u come probabilità che un certo nodo non appartenga alla giant component. Chiaramente se $u = 1$ non esiste giant component mentre se $u < 1$ esiste giant component.
Consideriamo poi i, un vertice che sappiamo non appartenere alla giant component.
Per ogni nodo $j \neq i$ all'interno del grafo vi sono due casistiche :

1. i non è connesso a j
2. i è connesso a j e questo implica che j non appartenga alla giant component

Per le due casistiche troviamo la loro probabilità corrispondente:

1. $1 - p$: probabilità che non vi sia un legame.
2. $p \cdot u$: p è la probabilità che vi sia un legame moltiplicata per u, probabilità che j non appartenga alla giant component.

Pertanto la probabilità di non essere connessi alla giant component tramite j è la somma delle due casistiche ovvero:

$$(1 - p) + (p \cdot u)$$

Se ne deduce che la probabilità di non essere connessi alla giant component tramite nessuno degli altri $n - 1$ nodi sia

$$u = ((1 - p) + (p \cdot u))^{n-1}$$

Riscrivendo tale equazioni in termini di $p = \dfrac{c}{n-1}$ abbiamo:

$$u = \left[1 - \frac{c}{n-1}(1 - u)\right]^{n-1}$$

Usando i logaritmi ad entrambi i lati abbiamo:

$$\ln(u) = (n - 1) \cdot \ln\left[1 - \frac{c}{n-1}(1 - u)\right]$$

Che per le serie di Taylor diventa:

$$\ln(u) \approx -(n-1)\frac{c}{n-1}(1-u) = -c(1-u)$$

Quindi usando l'esponenziale ad entrambi i membri otteniamo:

$$u = e^{-c(1-u)}$$

Ma se u è la probabilità che un elemento non sia nella giant component, allora $S = 1-u$ è la probabilità che lo sia, dunque possiamo anche vedere la formula da questo punto di vista:

$$S = 1 - e^{-cS} \tag{6.8}$$

Questa equazione ci da la dimensione della giant component come frazione della rete per qualsiasi valore del grado medio c.

Putroppo, nonostante l'equazione appaia abbastanza semplice non ha una soluzione chiusa. Possiamo però cercare di intravedere come si comporti tramite una rappresentazione grafica. S può prendere valori compresi fra 0 e 1 quindi

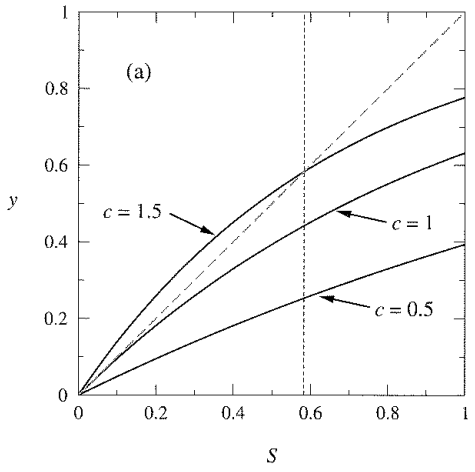


Figura 6.1: Valore di S per diversi valori di c.

solo quella parte è mostrata sul grafico. La linea diagonale tratteggiata mostra $y = S$. Quando la linea e la curva si intersecano abbiamo una soluzione: come vediamo è possibile avere una o due soluzioni. Per c piccolo abbiamo solo una soluzione in 0 e questo significa che non c'è giant component. Quando invece c è sufficientemente alto (circa 1.5) allora abbiamo due soluzioni, quella in 0 e quella in 0.6: c'è giant component.

6.2.7 Diametro medio della rete

Le reti random alla Erdös-Rényi sono reti dette *tree-like* ovvero si sviluppano con una forma simile a quella di un albero. Ciò significa che il clustering tende

a 0 quando c è costante (e questo lo avevamo già appurato).
Il numero di vertici a distanza ℓ è del tipo c^ℓ, ovvero partendo da un nodo a caso abbiamo un diametro medio ℓ per cui $c^\ell \approx n$ che passando ai logaritmi ci da $\ell \approx \frac{\ln n}{\ln c}$ cioè il diametro *cresce logaritmicamente*.

6.2.8 Riflessioni sul modello

Del modello Erdös-Rényi abbiamo capito che:

- La distrubuzione di grado è Poissioniana (a coda leggera), dunque non ci sono hub.
- La relazione fra n e il diametro è logaritmica.
- Il coefficiente di clustering è basso, non c'è giant component.

La domanda a cui è interessante rispondere è la seguente: questo modello è rappresentativo del mondo reale? Ovvero nella realtà le reti che studiamo rispecchiano queste tre caratteristiche? La risposta, come già accennato, è no (o almeno non completamente). La crescita logaritmica del diametro è abbastanza realistica ma le altre due caratteristiche invece sono raramente o mai riscontrabili nelle reti reali.
Perché dunque studiare questi modelli così poco realistici? I modelli random sono un buon *metro di paragone*. Se ad esempio notiamo una qualche particolarità in una rete possiamo confrontarla con una rete randomica: se il fenomeno si ripresenta allora tale fenomeno non era così interessante (infatti è generato dal caso, mentre noi stiamo tendenzialmente interessati ai fenomeni causali tangibili per cui una rete si comporta in una certa maniera).

6.3 Modelli a distribuzione generale dei nodi

Per risolvere il problema della distribuzione di Poisson (non realistica), sono state sviluppate alternative a questo modello (di cui noi accenniamo solo in questo paragrafo).
In sostanza si impone alla rete una distribuzione di grado creata a tavolino, così da poterne adoperare una più realistica.
Questa famiglia di modelli soffre comunque ancora del problema del basso coefficiente di clustering.
Tratteremo approfonditamente il modello di Price e poi faremo alcuni cenni al modello Barabàsi-Albert.

6.3.1 Un modello non statico

Il modello di Price è un modello *dinamico*, che prevede nella sua definizione matematica l'aggiunta di nuovi nodi ad un grafo. La dinamicità è un elemento

fondamentale nella realtà, moltissime reti sono risultato di una evoluzione nel tempo e non nascono così, dal nulla.
Il modello di Price è stato sviluppato per cercare di rappresentare una rete reale in particolare, quella delle citazioni scientifiche.
È necessario precisare che vi sono molte reti diversi di citazioni scientifiche, differenziate dall'argomento scientifico trattato: si notano infatti delle differenze non da poco già fra queste reti nonostante finiscano tutte sotto il cappello delle 'reti di citazioni'.
Questo modello prevede una rete *diretta* e rigorosamente *aciclica*. La rete è diretta poiché vi sono archi entranti/uscenti (quando un paper cita un altro paper si crea un arco che dal primo punta al secondo) e aciclica poiché è impossibile che un paper citi un paper dal futuro.
La distribuzione di grado entrante è una *power-law*, mentre invece la distribuzione per il grado uscente (paper citati) è rappresentabile da una *campana* Gaussiana poiché mediamente si citeranno dai 10 ai 30 paper, molto raramente ne avremo uno con 100 citazioni e sostanzialmente mai ne avremo uno da 10.000.
Chiaramente saremo più interessati allo studio della power-law, quindi dell'*in-degree*.

6.3.2 L'idea del preferential attachment

Quando un articolo ne cita un altro la scelta della citazione non è casuale ma si tenderà a citare un articolo rilevante ed autoritario in merito all'argomento di cui si sta scrivendo: questo è un passo notevole rispetto alla casualità portata dal modello random.
Esiste quindi una sorta di equivalenza tacita per cui un articolo molto citato (con in-degree alto) è automaticamente autoritario e quindi possiamo aspettarci che venga citato ulteriormente nel tempo a seguire dai nuovi articoli.
Questo principio è detto *preferential attachment*, ovvero il fatto che al suo arrivo nella rete un nuovo nodo non sceglierà a caso fra tutti i nodi presenti chi citare ma sceglierà tendenzialmente un nodo con grado già alto (molto citato).

6.3.3 Master equation

Per descrivere matematicamente i concetti che abbiamo sopra esposto, Price sviluppa una *master equation* che ci può far capire cosa accade alla rete quando arriva un nuovo nodo.

Calcoli preliminari

Chiamiamo:

- q_i: l'in-degree del vertice i.
- $P_q(n)$: la frazione dei vertici della rete con grado q (su un totale di n vertici).
- a: un numero di citazioni 'gratuite' che vengono assegnate ad un nodo in arrivo. Si noti che questa costante è solo un escamotage matematico per far funzionare il sistema probabilistico: un nuovo nodo non riceve effettivamente a archi, ma dato che per definizione 'un nodo è appetibile in base al numero di citazioni che ha', se i nuovi nodi avessero zero citazioni non potrebbero mai essere citati. Nel caso di modello di Price, $a = 1$.

La probabilità che un nuovo nodo citi un particolare nodo i è quindi proporzionale a $q_i + a$. Giacché queste citazioni andranno a 'qualche' paper, è saggio normalizzare in modo che la somma di queste probabilità su tutti gli i sia 1. Otteniamo quindi:

$$\frac{q_i + a}{\sum_i (q_i + a)} = \frac{q_i + a}{n\langle q \rangle + na} = \frac{q_i + a}{n(c + a)} \tag{6.9}$$

dove abbiamo scritto l'in-degree medio come $\langle q \rangle = \frac{1}{n} \sum_i q_i$. Nella seconda equazione abbiamo quindi fatto un ulteriore passo grazie all'equivalenza $\langle q \rangle = c$. Tale equivalenza è legittima poiché l'out-degree di un network è per definizione c e l'in-degree e l'out-degree medi di un grafo sono uguali (non lo dimostriamo qui, ma lo diamo per acquisito).

Ogni nuovo nodo cita mediamente c paper, da ciò se ne deduce che il numero atteso di nuove citazioni a vertici i sarà c volte la probabilità espressa in Eq. 6.9.
Vi saranno $np_q(n)$ vertici con degree q nella rete prima dell'arrivo del nuovo nodo, dunque il numero atteso di nuove citazioni a tutti i vertici attualmente di grado q risulta essere la moltiplicazione fra $np_q(n)$ e c volte Eq. 6.9:

$$np_q(n) \times c \times \frac{q + a}{n(c + a)} = \frac{c(q + a)}{c + a} p_q(n) \tag{6.10}$$

Stesura dell'equazione

Ora siamo in grado di scrivere la master equation per descrivere l'evoluzione dell'in-degree.
All'arrivo di un nuovo nodo, potranno diventare di grado q solo i nodi che erano di grado $q - 1$ e vengono citati, allo stesso modo i nodi attualmente di grado q che verranno citati diventeranno di grado $q + 1$. Abbiamo quindi un contributo positivo dato dai nodi attualmente di grado $q - 1$ che diventeranno

di grado q e un contributo negativo dato dai nodi attualmente di grado q che diventeranno di grado $q+1$. Possiamo calcolare la quantità attesa di questi due gruppi di nodi grazie alla Eq. 6.10.
I nodi che da $q-1$ diventeranno di grado q sono:

$$\frac{c(q-1+a)}{c+a}p_{q-1}(n) \tag{6.11}$$

I nodi che da q diventeranno di grado $q+1$ sono:

$$\frac{c(q+a)}{c+a}p_q(n) \tag{6.12}$$

Il numero di vertici con in-degree q che si troveranno nella rete in seguito all'arrivo di un nuovo nodo saranno dunque quelli attualmente di grado q, più quelli che diventeranno di grado q (Eq. 6.11), meno quelli che perderanno il grado q (Eq. 6.12).

$$(n+1)p_q(n+1) = np_q(n) + \frac{c(q-1+a)}{c+a}p_{q-1}(n) - \frac{c(q+a)}{c+a}p_q(n) \tag{6.13}$$

L'equazione sopra si applica per tutti i valori di q tranne per $q=0$. Quando $q=0$ non ci sono vertici di grado minore che possano diventare di $q=0$ quindi il secondo termine dell'Eq. 6.13 non appare. D'altronde, otteniamo un nuovo vertice di grado 0 ogni volta che aggiungiamo un nodo (si ricorda che il grado entrante a è solo un escamotage matematico) quindi abbiamo un $+1$. Perciò l'equazione risulta essere:

$$(n+1)p_0(n+1) = np_0(n) - \frac{ca}{c+a}p_0(n) + 1 \tag{6.14}$$

Calcoliamo il limite per $n \to \infty$ (usando come abbreviazione $p_q = p_q(\infty)$) e otteniamo rispettivamente per $q \geq 1$ e $q = 0$:

$$p_q = \frac{c}{c+a}[(q-a+a)p_{q-1} - (q+a)P_q] \tag{6.15}$$

$$p_0 = 1 - \frac{ca}{c+a}p_0 = \frac{1+a/c}{a+1+a/c} \tag{6.16}$$

Mentre di p_0 abbiamo già scritto una versione esplicita, possiamo lavorare su p_q in modo da ottenere questa scrittura ricorsiva:

$$p_q = \frac{q+a-1}{q+a+1+a/c}p_{q-1} \tag{6.17}$$

La soluzione di questa equazione ricorsiva (che non svolgiamo qui) risulta essere:

$$P_q \approx q^{-\alpha} \tag{6.18}$$

dove, quando $q \gg a$, $\alpha = 2 + \frac{a}{c}$.
È una power-law!

6.3.4 Cenni al modello Barabàsi-Albert

Il modello di Barabàsi-Albert è un modello molto simile a quello di Price ma presenta due differenze:

- $a = 0$.
- Il grafo non è orientato (quindi usiamo il grado k al posto del $q_i n$).

In sostanza si unificano i concetti di grado entrante ed uscente. Manteniamo dunque tutte le caratteristiche di dinamicità e di preferential attachment. Il modello è stato sviluppato per lo studio della rete del Web (come le pagine internet si linkano fra loro).
Possiamo ri-effettuare tutte le riflessioni già fatte per il modello di Price e ottenere anche qui una master equation. Il risultato ottenuto è:

$$P_k \approx k^{-}3$$

Non ci sorprende, anche qui abbiamo una power-law e quindi ritroviamo il concetto di scalabilità.

6.3.5 Power-law e limiti di sviluppo

Si noti che vi sono reti che per motivi intrinsechi non possono essere power-law.
Pensiamo ad esempio a reti i cui nodi soffrono di *aging* (la rete degli attori): attori che precedentemente erano hub diventeranno, man mano, meno importanti poiché non potranno più recitare (sono morti oppure non sono più in attività lavorativi) e quindi ci sarà un continuo riciclo nella rete.
Un altro caso in cui la distribuzione power-law non si può sviluppare è quello in cui la connessione a nodi con degree già alto diventa costosa. Pensiamo ad un sistema di aereoporti: aggiungere rotte a un aereoporto già saturo è evidentemente un costo (in termini organizzativi e di sicurezza), dunque è più semplice effettuare rotte differenziate su altri nodi piuttosto che sugli areoporti già saturi di voli.
Collegato a questo fenomeno c'è il famoso caso della rete gnutella. Nel periodo in cui le power-law avevano molto 'successo' si tendeva a trovare power-law ovunque (magari alla ricerca di una pubblicazione facile). Un analisi della rete gnutella (peer-to-peer) sembrava aver rivelato, anche qui, una power-law.
Tale studio si rivelerà poi erroneo (l'analisi di rete era effettuata in maniera impropria), ma il fatto che una rete del genere non potesse essere power-law era già intuibile: il numero di archi (connessioni) che possono partire da un calcolatore è finito.

6.3.6 Il vertex copying

Abbiamo parlato del modello di Price e di come il preferential attachment influenzi i legami verso i nodi pre-esistenti di una rete all'arrivo di un nuovo vertice.
Un altro approccio è quello del vertex copying: un nuovo paper non si limiterà a citare i paper precedenti ma probabilmente copierà le citazioni da un altro nodo!
Questo fatto viene formalizzato tramite una probabilità P per cui: quando un nuovo nodo è annesso alla rete viene scelto uno fra gli n nodi presenti e vengono prese in considerazioni le sue c citazioni. Per ognuna di esse, con probabilità $P = \gamma$ copieremo la citazione mentre con probabilità $1 - P$ effettueremo una citazione random fra gli altri n nodi (compreso quello da cui stiamo copiando).
Anche in questo caso potremmo ottenere una master equation; la correlazione interessante che risulta è che l'α di Pierce e la probabilità $P = \gamma$ risultano legate come segue:

$$\alpha = +\frac{1}{\gamma}$$

Un altro esempio di vertex copying (che citeremo spesso in seguito) è la rete delle proteine: ogni nuova proteina copia elementi da proteine pre-esistenti seguendo uno schema ben rappresentato dal modello vertex copying.

6.3.7 Reti strategiche

Altrettanto interessante caso di studio sono le reti che, all'annessione di nuovi nodi, si basano su un concetto strategico per definire i c archi. Pensiamo alle reti degli *autonomous systems* che effettuano nuovi collegamenti basandosi sull'*ottimizzazione*.
Allo stesso modo possiamo pensare alle reti che rappresentano giochi (in riferimento alla teoria dei giochi): ogni giocatore segue una strategia.

6.3.8 Study case: diffusione di una patologia

Tocchiamo con mano quanto appena studiato esaminando uno script in NetLogo raffigurante una rete di Erdös-Rényi, applicata allo study case della diffusione di una patologia.
Possiamo esprimere il degree e la probabilità p con cui la malattia si diffonde. La diffusione è rappresentata in modo molto semplice: se un nodo è infetto, infetta con probabilità p (visibile in figura) ognuno dei suoi nodi adiacenti.
Vediamo la creazione di una rete con basso grado medio:

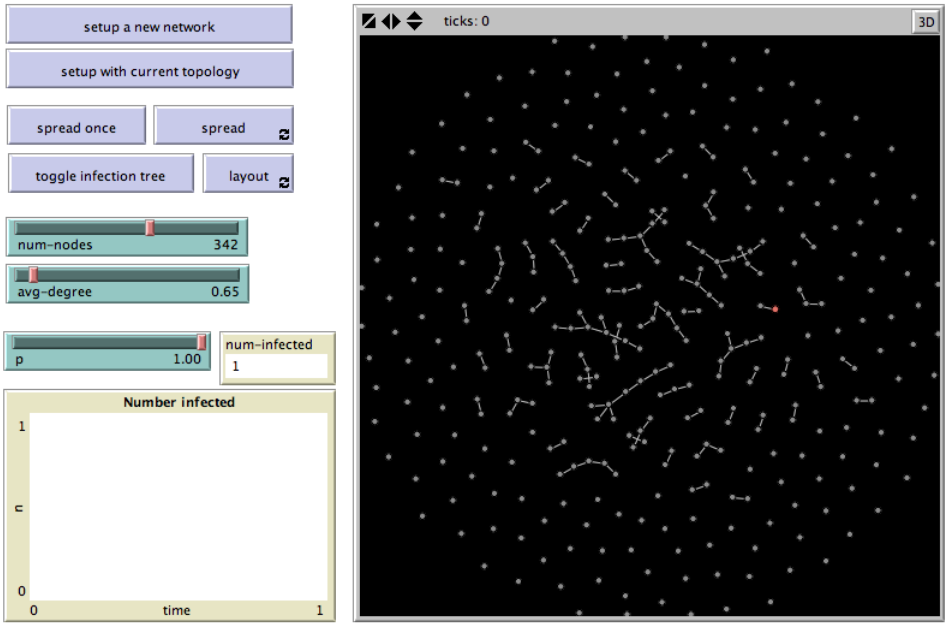


Figura 6.2: Una rete con basso grado medio (0.65).

Vediamo in rosso il nodo infettato. Ora rilasciamo l'infezione e vediamo questo risultato: Gli infettati sono solamente 2 nonostante p fosse al massimo (1).

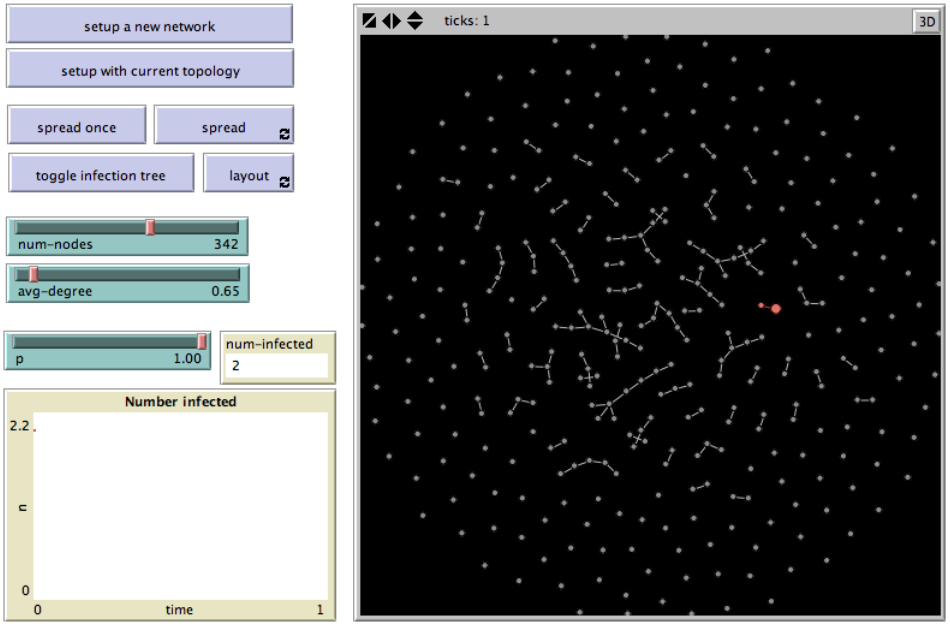


Figura 6.3: Ci sono pochi infetti a fronte di un basso grado medio.

Se ora aumentiamo il grado, otteniamo questo tipo di infezione:

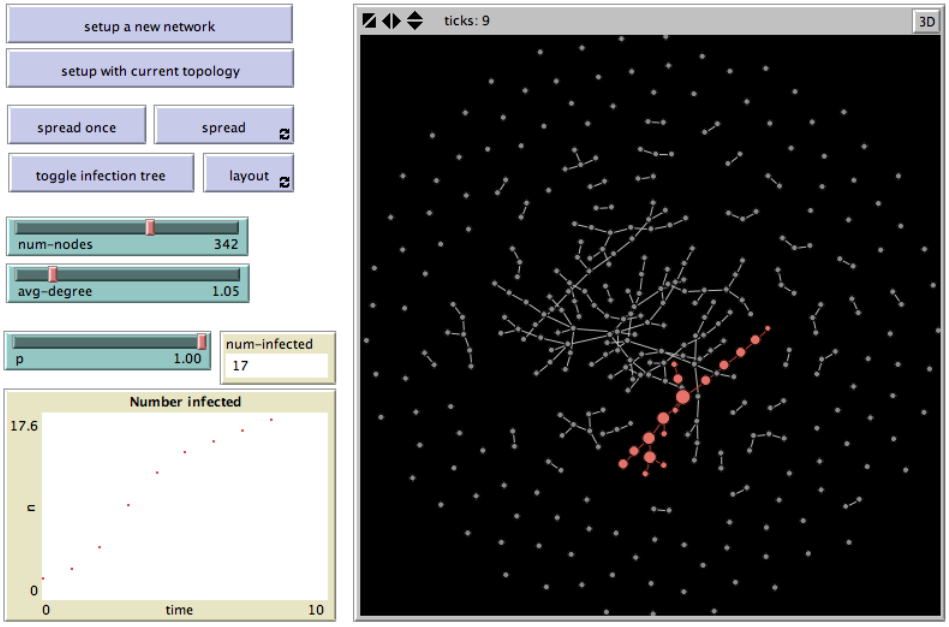


Figura 6.4: Ci sono più infettati con un grado più alto.

E aumentandolo ancora:

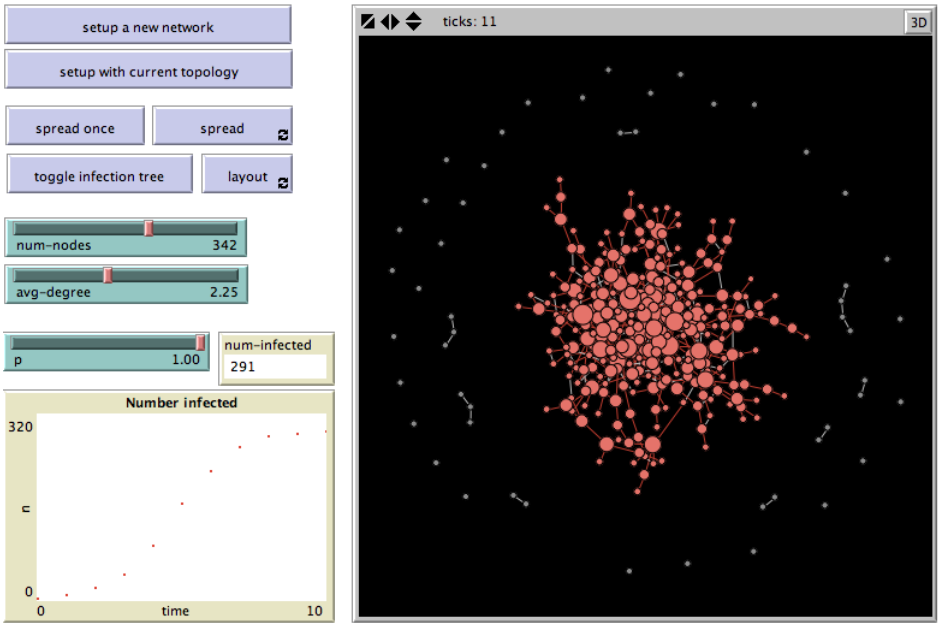


Figura 6.5: Quasi tutta la rete è stata infettata.

Come vediamo, quando c'è una giant component (più c è prossimo a 1.5 più la

componente aumenta come avevamo appurato nella Sezione 6.2.6) quasi tutta la rete è infettata. C'è quindi una grossa dipendenza fra 'quanto le persone si conoscono' e la diffusione dell'agente patogeno.
È chiaro che anche la probabilità p influisca sullo sviluppo della malattia, ma solo in termini di tempo: con p più bassa il risultato sarà molto simile ma con uno sviluppo più lento.

6.4 Modello small world

Il concetto di modello small world viene toccato per la prima volta da Milgram che col suo esperimento scopre i famosi *sei gradi di separazione*. Tale concetto implica il fatto che nella rete si ha un corto diametro. Le reti small word per antonomasia sono le reti sociali.

6.4.1 Lo studio dei motifs

Iniziamo a ragionare sui motifs e su quanto la loro presenza possa dirci qualcosa.
Un motifs è una certa combinazione di archi e vertici che può essere ripetuta all'interno di uno stesso grafo. Possiamo immaginarlo come un pattern che, se ripetuto molte volte in una rete, può essere considerato un elemento interessante. La presenza di taluni motifs a tre vertici può indicare ad esempio la presenza di alto clustering.
Lo studio effettivo dei motifs si effettua misurando la differenza di motifs fra una rete randomica (Erdös-Rényi) e la rete reale. La rete randomica viene generata mantenendo lo stesso grado dei vertici di quella reale. Quindi se esaminassimo una rete reale del genere, individuandone i motifs che identificano il clustering:

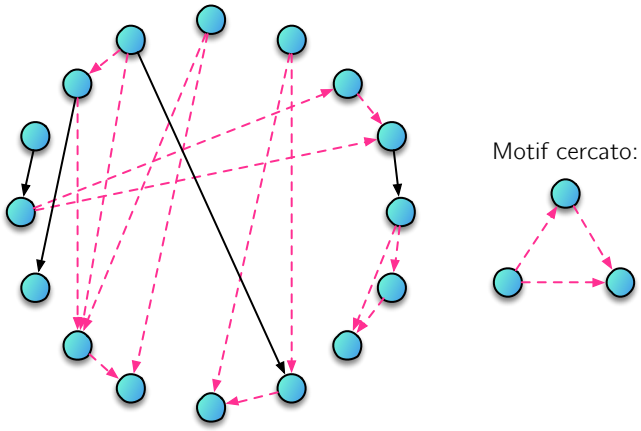


Figura 6.6: Motifs trovati nella rete reale.

Dovremmo poi studiarne altre versioni generate randomicamente: In questo

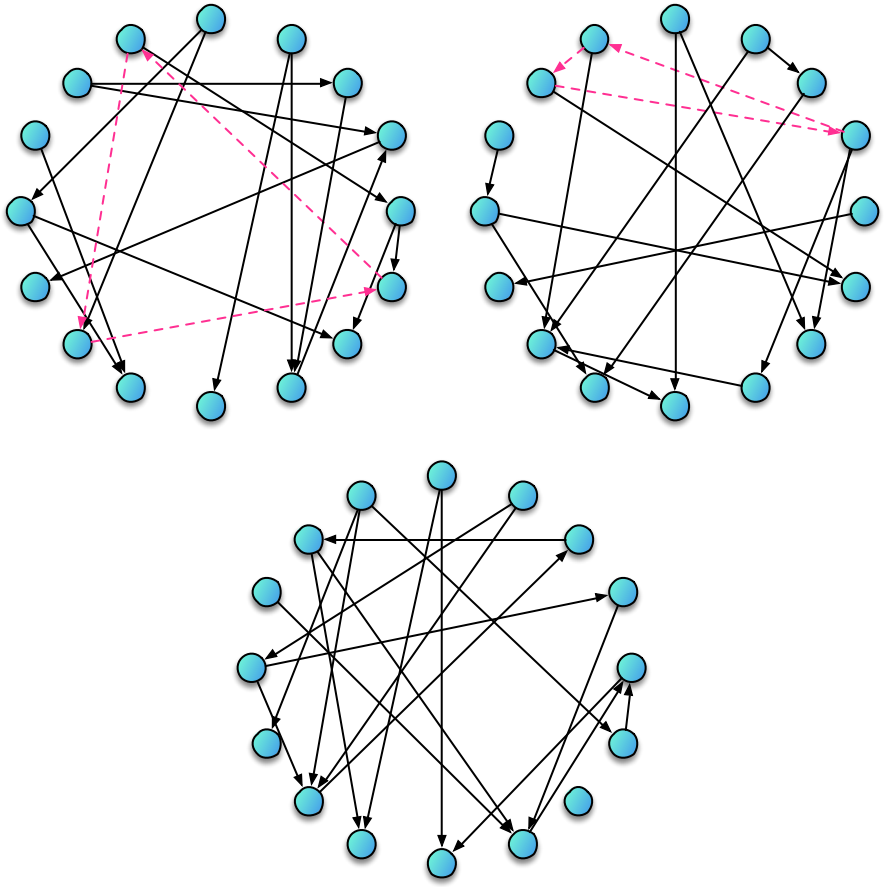


Figura 6.7: Motifs trovati nei grafi random (stesso grado dell'originale).

caso il fenomeno è interessante poiché ci sono moltissimi motifs nel grafo originale e pochissimi in quello random.

6.4.2 Il modello (Watts-Strogatz)

Il rapporto fra path e clustering

Prima di parlare del modello che rappresenta le reti small world è necessario capire quali siano le caratteristiche che vogliamo mimare. Le reti sociali hanno *alto clustering*, ovvero la maggiorparte dei rapporti di amicizia è sovrapposto con presenza correlata di omofilia.
L'altra caratteristica interessante è che il diametro cresce logartimicamente, dunque ci sono path fra i nodi molto corti. Il nostro intento è quindi quello di creare una rete che contemporaneamente abbia alto clustering e un path medio

molto breve. Watts precisamente delinea quattro caratteristiche identificanti una rete small world:

- La rete deve essere grande (milioni di nodi).
- La rete deve essere sparsa.
- La rete deve essere decentralizzata (non molti hub).
- La rete deve essere molto clusterizzata.

Rewiring

L'idea base sviluppata da Watts e Strogatz è quella di partire da un reticolato con clustering massimo e poi modificare alcuni link nella speranza di ottenere un average path sufficientemente basso così da riprodurre le situazioni reali delle reti small world. L'operazione di ri-connesione degli archi è detta *rewiring* ed è guidata da una probabilità p detta *probabilità di rewiring*.
Ogni arco verrà ri-collegato con un nodo random della rete con probabilità p. Ovviamente la definizione della probabilità è fondamentale per preservare il clustering. Il reticolato regolare ha massimo clustering ma ha un average path

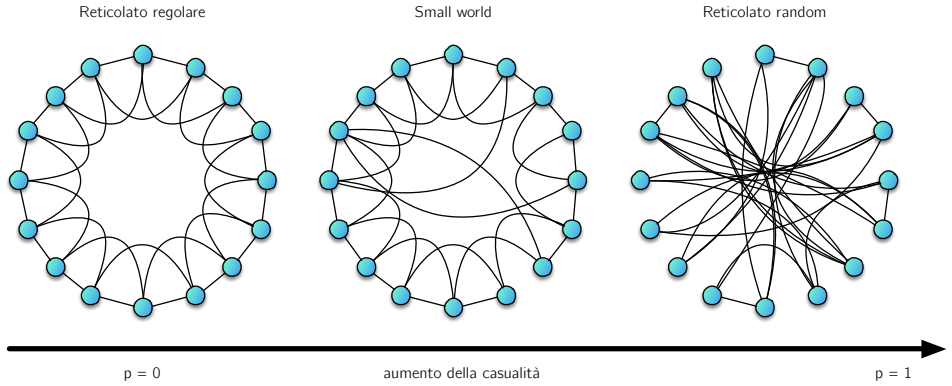


Figura 6.8: Cambiamento del reticolato in base a p.

molto lungo. Con un aumento minimo della p possiamo staccare *qualche arco* e ottenere così path sufficientemente corti pur mantenendo un alto clustering. Ecco le reti small world.
Possiamo più formalmente indicare il rapporto fra le due grandezze tramite la Figura 6.9. Esiste dunque un punto di transizione che ci permette di mantenere il clustering e di avere anche uno short path.

Si noti che esiste una tecnica alternativa per generare reti small world e consiste nell'aggiungere archi ad un reticolato a massimo clustering invece di modificare i già presenti.

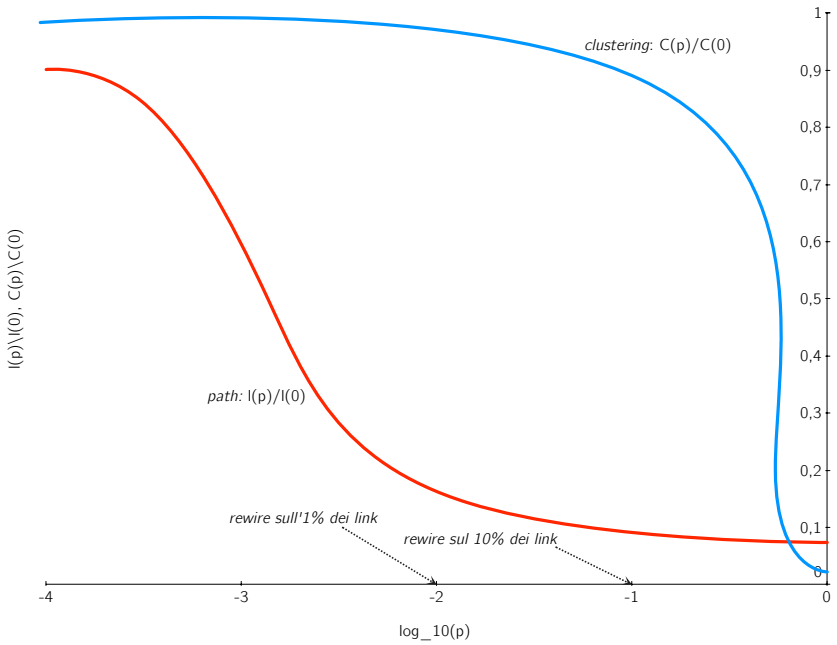


Figura 6.9: Confronto fra path e clustering.

6.4.3 Grafi random per determinare situazioni small world

Per capire se esiste una situazione small world, come abbiamo detto, si effettua un confronto fra la rete reale e una rete random (Erdös-Rényi) di pari distribuzione di grado. Vediamo dunque alcuni risultati rinvenuti su rete reali (SP sta per shortest path, C per clustering ed Rand è il calcolo su grafo random equivalente).

Rete	Nodi	$\langle$ SP $\rangle$	$\langle$ SP $\rangle$ (Rand)	$\langle$ C $\rangle$	$\langle$ C $\rangle$ (Rand)
Attori	225,226	3.65	2.99	0.79	0.00027
Autori	1,520,251	4.60	4.91	0.56	1.8×10^{-4}
E. Coli	282	2.90	3.04	0.32	0.026
C. Elegans	302	2.65	2.25	0.28	0.05

La penultima rete riguarda una rete di diffusione del batterio Escherichia Coli mentre l'ultima rete riguarda una rappresentazione della rete neurale di un particolare verme, il Caenorhabditis Elegans.
Da quanto analizzato vediamo che il clustering è un elemento estremamente 'non casuale' in questo tipo di reti. Le reti tecnologiche come abbiamo già detto più volte hanno meno clustering.

6.5 La navigabilità di Kleinberg

L'esperimento di Milgram è stato fondamentale per determinare il fatto che nelle reti esistesse una struttura tale da permettere l'esistenza dei famosi 'sei gradi di separazione'. Kelinberg però vede qualcosa di ulterioremente interessante nascosto sotto l'esperimento di Milgram: un'idea algoritmica di fondo. Esiste un meccanismo semplice col quale i messaggi raggiungono la destinazione, guidati da informazioni locali e non globali (quindi non si effettua una analisi della rete e si studia lo shortest path per inviare il messaggio seguendo tale path, ma si sfrutta qualcos'altro).
Ogni nodo invia dunque il messaggio ad un suo vicino, sfruttando una tecnica greedy: ad ogni passo si cerca di avvicinarsi il più possibile all'obiettivo senza avere una visione globale del problema.
Dal punto di vista algoritmico questo approccio è molto più semplice poiché le informazioni necessarie per sfruttare la *navigabilità* sono pochissime e locali (facili da reperire).

6.5.1 Ottenere la navigabilità

Come può il nostro modello simulare la navigabilità? Partiamo anche questa volta da un reticolo regolare, dove ogni nodo è collegato coi suoi quattro vicini. Dopodiché per ogni nodo aggiungiamo un link con un nodo casuale non locale con probabilità p inversamente proporzionale alla distanza fra i due nodi e ad un fattore r. Quindi $p \approx \frac{1}{d}^r$.

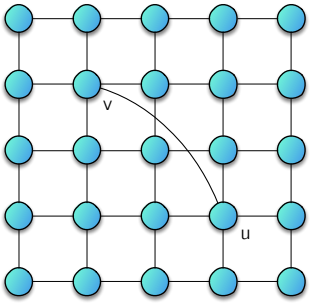


Figura 6.10: Fra u e v viene messo un link con $p = \approx \frac{1}{d}^r$.

Notiamo che quando $r = 0$, ogni nodo ha un nuovo link con un nodo casuale e il nostro grafo si trasforma in un grafo random, questo fa sì che l'agire greedy sia in realtà non così vantaggioso (si torna spesso indietro sui propri passi perdendo così di vista il goal). È dunque evidente che non ci vogliano *molti link*, ma che vi sia un valore di r giusto per far sì che i link siano utili e creino la nostra desiderata navigabilità.

È dimostrato che quel valore di r è 2. Perciò quando $p = \frac{1}{d^2}$ abbiamo massima navigabilità.

6.6 Epidermiology

L'epidermiologia è lo studio scientifico di fattori che affliggono la salute di una popolazione. Durante il nostro studio ci concentreremo su epidemie di *malattie infettive*.

6.6.1 La malattia

Una certa malattia è definibile come il connubio di tre caratteristiche:

- Agente infettivo (capacità di infezione, agente patogeno, come viene immunizzata la malattia, probabilità di sopravvivenza).
- Ospite (età, sesso, genotipo, comportamento, stato nutrizionale, stato di salute).
- Ambiente (condizioni climatiche, posizione geografica, qualità dell'aria, qualità del cibo).

6.6.2 Le reti di contatti: modelli epidemici

La malattia si diffonde all'interno della rete tramite il contatto fra diverse entità. Tale contatto può essere di natura diversa, ad esempio:

- Trasmissione per via area (influenza o tubercolosi).
- Trasmissione per tocco (parassiti).
- Trasmissione sessuale (HIV).

È evidente che per studiare la diffusione della malattia si renda necessario rappresentare la rete di contatti. Tale rete può rappresentare interconnessioni fra singoli individui, fra gruppi (ad esempio una sotto-popolazione in un certo range di età) oppure fra popolazioni vere e proprie (ogni nodo rappresenta una città o un paese).

Si vengono quindi a delinare i cosiddetti *modelli epidemici*. Ve ne sono di diversi tipi, man mano più realistici (e quindi complessi). Vediamone una carrellata.

Homogeneous mixing models Viene assunto che gli individui interagiscano fra loro in maniera totalmente random e che siano tutti collegati fra loro.

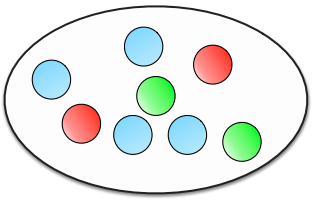


Figura 6.11: Rappresentazione di un modello homogeneous mixing.

Social structure models Gli individui sono categorizzati secondo una certa caratteristica demografica (ad esempio età oppure sesso).

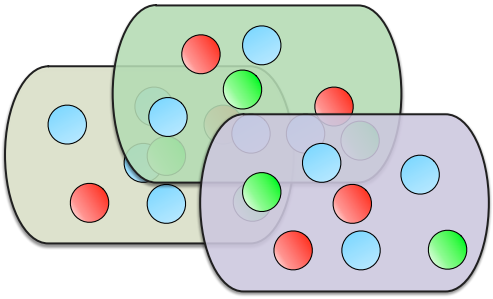


Figura 6.12: Rappresentazione di un modello social structure.

Contact network models Vengono rappresentati le interazioni sociali fra individui con l'intento di delineare possibili path di diffusione della malattia.

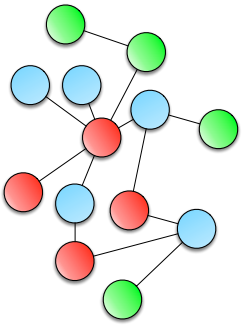


Figura 6.13: Rappresentazione di un modello contact network.

Multi-scale models Si interconnettono più modelli omogenei per poter rappresentare connessioni fra sottopopolazioni (rappresentate quindi al loro interno in modo omogeneo).

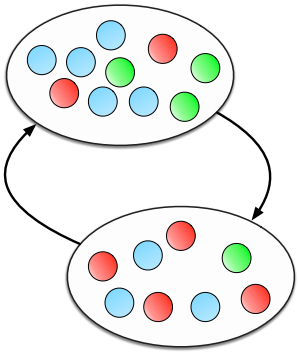


Figura 6.14: Rappresentazione di un modello multi-scale.

Agent- based models Si tenta di rappresentare in modo dettagliato tutte le azioni compiute da un individuo. Chiaramente questo modello è applicabile se si prende in considerazione una piccola area di movimento dell'individuo (ad esempio la città).

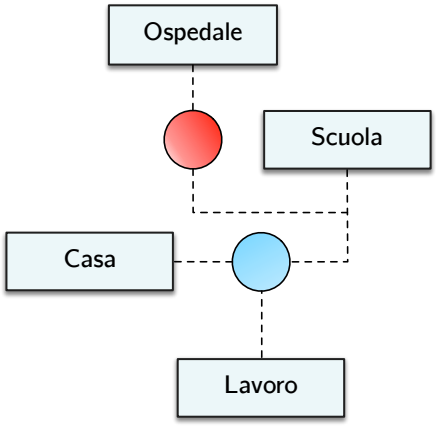


Figura 6.15: Rappresentazione di un modello agent-based.

6.6.3 Il modello SI

Il modello *SI* è un semplice modello epidemico che permette la rappresentazione di una popolazione e la diffusione di una malattia in questo. Il modello si basa su due semplici regole:

- In qualsiasi istante un individuo può essere suscettibile (S, *susceptible*) oppure infetto (I, *ill*). Se un individuo è infetto allora è anche infettivo.
- Solo gli individui suscettibili (quindi $\in S$) possono essere infettati.

Il modello è davvero semplice poiché sono poste molte assunzioni: la popolazione è costante, non c'è latenza nella trasmissione dell'agente patogeno, non c'è differenza di sorta fra gli individui (sesso, età, occupazione, ecc.), non esistono ospedali né medicine e non vi sono cambiamenti nell'ambiente circostante.
Viene poi assunto il fatto che ogni individuo abbia pari probabilità di entrare in contatto con qualunque altro individuo del network (e questa è evidentemente una condizione poco realistica). Se ne deduce quindi che la rete sia completamente connessa.
Sfruttiamo due funzioni per descrivere la situazione in un certo tempo t:

- $S(t)$ è il numero di individui suscettibili.
- $X(t)$ è il numero di individui infetti.

Inoltre ogni individuo ha mediamente, in ogni istante di tempo, un numero β di contatti con altri individui. Il numero di individui infetti aumenterà quindi nel momento in cui individui suscettibili contrarranno la malattia da altri individui già infetti.
Riassumendo abbiamo che con una popolazione di n membri $\frac{S}{n}$ è la probabilità media che una persona sia suscettibile, dunque una persona infetta (come ogni altra persona) entra in contatto (sempre mediamente) con $\beta \cdot \frac{S}{n}$ persone suscettibili per ogni unità di tempo.
Perciò, se ci sono X infetti in totale avremo una probabilità di nuova infezione di:

$$\frac{dX}{dt} = \beta \frac{SX}{n}$$

Ne deduciamo quindi che il numero di individui suscettibili scenda in maniera inversa cioè:

$$\frac{dS}{dt} = -\beta \frac{SX}{n}$$

Per semplificare la scrittura potremmo considerare:

$$\begin{aligned} s &= S/n \\ x &= X/n \end{aligned} \tag{6.19}$$

E quindi riscrivere le due formule come:

$$\frac{ds}{dt} = -\beta sx \qquad\qquad \frac{dx}{dt} = \beta sx$$

La funzione di infezione

Abbiamo ora una equazione differenziale che descrive l'andamento delle infezioni nel tempo. Ma sapendo che:

$$S + X = n \Rightarrow s + x = 1$$

possiamo facilmente effettuare la sostituzione:

$$\frac{dx}{dt} = \beta(1 - x)x$$

e quindi risolvere l'equazione differenziale ottenendo la funzione che governa il numero di individui infetti:

$$x(t) = \frac{x_0 e^{\beta t}}{1 - x_0 + x_0 e^{\beta t}}$$

dove x_0 è il numero di infetti x all'istante 0.
Il comportamento della funzione risulta essere il seguente:

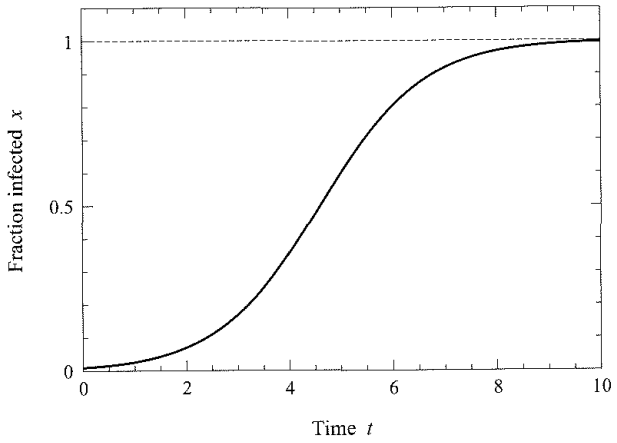


Figura 6.16: Andamento degli infetti nel tempo ($x(t)$).

6.6.4 Il modello SIR

Il modello *SIR* è leggermente più complesso del modello *SI* poiché introduce un nuovo stato: il *recovered*, rappresentato dalla funzione $R(t)$. Si aggiunge quindi dinamicità (e realisticità) al modello avendo un totale di tre stati (susceptible, ill, recovered). Il modello si sviluppa in due stadi:

- Nello stadio iniziale gli individui suscettibili vengono infettati nel caso in cui entrino in contatto con un individuo infetto e si ha che due individui si incontrano con probabilità β (quindi esattamente come nel modello *SI*).

- Nel secondo stadio abbiamo che gli individui malati recuperano (o muoiono) con probabilità γ (con tempo medio di infezione pari a $\frac{1}{\gamma}$).

La distribuzione del tempo di infezione è considerata esponenziale in questo modello anche se per malattie vere si delinea una distribuzione appena accentuata intorno ad un valore medio.
Chiariamo un concetto: quando un individuo transisce in uno stadio di recovered *non può più essere infettato*, quindi non torna in uno stato di suscettibilità. Abbiamo quindi la solita funzione che regola gli individui suscettibili (ricordiamo le semplificazioni fatte prima in 6.19):

$$\frac{ds}{dt} = -\beta sx$$

Per quanto riguarda la funzione che regola gli infetti dobbiamo considerare la presenza dei recuperati (che sarà il numero di infetti x per la probabilità che recuperino γ):

$$\frac{dx}{dt} = \beta sx - \gamma x$$

Abbiamo quindi la funzione che regola i recovered (anche qui $r = r/n$) pari a:

$$\frac{dr}{dt} = \gamma x$$

Infine, sappiamo che:

$$S + X + R = n \Rightarrow s + x + r = 1$$

Cioè la somma degli infetti, dei recovered e dei suscettibili è pari ad n. Andiamo a risolvere le equazioni differenziali e otteniamo questo grafico:

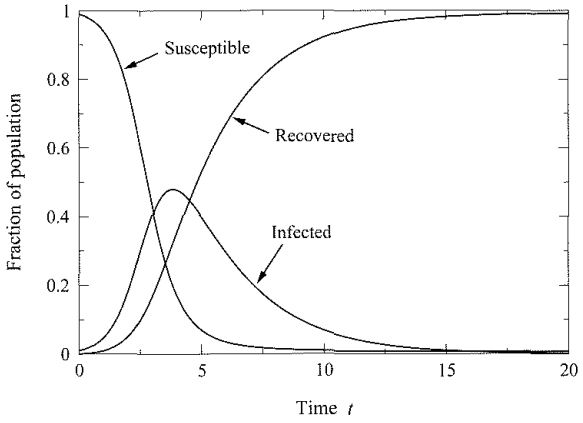


Figura 6.17: Andamento delle funzioni $s(t)$, $x(t)$ ed $r(t)$.

Grazie a questo grafico possiamo vedere che la frazione di individui suscettibili

decresce in maniera monotona tanto quanto la frazione di individui in stato di recovered aumenta in modo monotono.
Notiamo poi che la frazione di individui infetti inizialmente aumenta con l'aumentare degli infetti ma poi diminuisce con l'aumentare dei recovered fino ad arrivare a 0 con $t \to \infty$. Si noti comunque che il numero di suscettibili non va a zero ma si ferma appena sopra l'asse delle x (questo è spiegabile dal fatto che quando gli infetti arrivano a zero non esistono più suscettibili da infettare e quindi la malattia non potrà mai propagarsi attraverso gli ultimi suscettibili rimasti).

L'outbreak

Altro elemento fondamentale è il valore asintotico di r che ha una facile interpretazione pratica: è il numero*totale di individui che hanno contratto la malattia* durante il corso dell'epidemia (il cosiddetto *outbreak*). Possiamo trovare facilmente questo valore calcolando:

$$\frac{dr}{dt} = 0 \Rightarrow r = 1 - s_0 e^{-\beta \frac{r}{\gamma}}$$

Giacché tipicamente la condizione iniziale in cui la malattia si trova vede un solo infettato o comunque pochi c individui, impostiamo le nostre variabili di start come: $s_0 = 1 - c/n$, $x_0 = c/n$ e $r_0 = 0$ e quindi con $n \to \infty$ si ha un outbreak di:

$$r = 1 - e^{-\beta \frac{r}{\gamma}}$$

Relazioni con la giant component (epidemic transition)

Esiste una interessante correlazione con la giant component di un grafo Poissoniano. Se poniamo β/γ uguale al grado medio di un grafo random possiamo infatti ottenere informazioni importanti.
Per una grande popolazione (n grande) si ha che la dimensione dell'epidemia è funzione di β e γ. Più precisamente se $\dfrac{\beta}{\gamma} \leq 1$ (cioè $\beta \leq \gamma$) l'epidemia non si propaga (la spiegazione più intuitiva è il fatto che il numero di suscettibili che diventano infetti ad ogni istante t è minore del numero di infetti che diventano recovered in ogni istante t dunque la malattia non si può diffondere).
Capiamo dunque che la transizione fra un regime epidemico ed uno non-epidemico accade quando:

$$\beta = \gamma$$

Tale equivalenza è infatti detta *epidemic transition*.

Il basic reproduction number

Consideriamo il diffondersi di una malattia ai suoi primordi, quando ci sono solo pochi casi di infetti e il resto della popolazione è suscettibile (in gergo epider-

miologico si dice *naive population*). Consideriamo poi un certo individuo che viene infettato in questa fase primordiale dell'epidemia: il numero di persone a cui questo individuo trasmetterà la malattia viene definito *basic reproduction number* e lo rappresentiamo come R_0.
Per esempio con un valore $R_0 = 2$ abbiamo che ogni persona che contrae la malattia la trasmette mediamente ad altre due, ognuna delle quali la trasmetterà ad ulteriori due e così via. È intuibile che ogni tempo t ci porti ad un raddoppiamento degli infetti e quindi ad una crescita esponenziale. Al contrario, se avessimo avuto una $R_0 = \frac{1}{2}$ allora la malattia si sarebbe estinta esponenzialmente.
Il valore $R_0 = 1$ è il valore di mezzo che separa il comportamento di crescita da quello di decrescita dell'epidemia ed è detto *epidemic threshold*. Possiamo facilmente calcolare l'R_0 per il nostro modello.
Infatti se un individuo rimane infetto per un tempo τ allora il numero di persone con cui entrerà in contatto durante quel tempo sarà pari a $\beta\tau$. Ma definizione di R_0 è specificamente fondata sulla presenza di una naive population, dunque sotto queste condizioni $\beta\tau$ è anche il numero di persone che il nostro individuo infetterà (poiché con una naive population ogni individuo incontrato è suscettibile). Dunque se facciamo la media sulla distribuzione di τ otteniamo che (l'integrale deriva da altri ragionamenti che non abbiamo fatto):

$$R_0 = \beta\gamma \int_0^\infty \tau e^{-\gamma\tau} d\tau = \frac{\beta}{\gamma}$$

Abbiamo ottenuto lo stesso risultato di prima, infatti se $\beta = \gamma$ abbiamo che $R_0 = 1$.

6.6.5 Il modello SIS

Il modello SIS prevede i due stati che abbiamo già esaminato nel modello SI ma prevede la possibilità che un individuo malato possa tornare suscettibile (quindi la malattia non conferisce immunità). Definiamo come γ la possibilità di un infetto di tornare nello stato di suscettibilità, dunque abbiamo le equazioni differenziali:

$$\frac{ds}{dt} = \gamma x - \beta s x \qquad\qquad \frac{dx}{dt} = \beta s x - \gamma x$$

E come sempre:

$$s + x = 1$$

La soluzione della seconda equazione differenziale ci fornisce la funzione degli infetti nel tempo che risulta essere:

$$x(t) = x_0 \frac{\beta - \gamma e^{(\beta-\gamma)t}}{\beta - \gamma + \beta x_0 e^{(\beta-\gamma)t}}$$

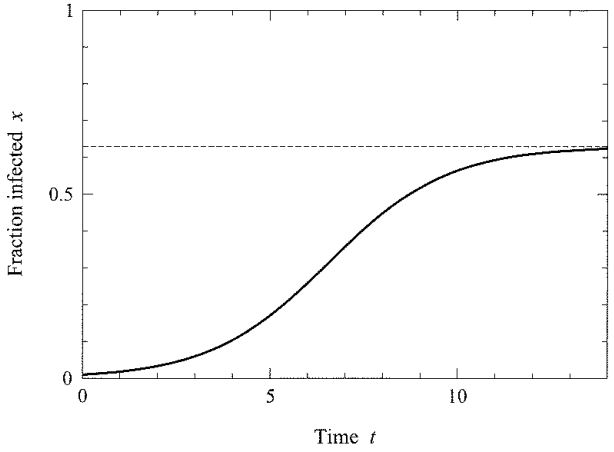


Figura 6.18: Andamento degli infetti nel tempo ($x(t)$).

Che possiamo visualizzare graficamente come:
Se abbiamo che $\beta > \gamma$ la curva prodotta è simile a quella che abbiamo già studiato nel modello *SI* ma con una importante differenza: la popolazione non potrà mai essere contagiata del tutto. Infatti con un tempo sufficientemente lungo possiamo vedere che il sistema si stabilizza in uno stato dove il numero di individui che sono infetti e quelli che recuperano sono esattamente uguali, dunque una buona fetta della popolazione (ma non tutta) è costantemente malata (si ha che alcuni individui guariscono ma altrettanti si ammalano nuovamente). In epidermiologia si parla di *endemic disease state*.
La quantità di infetti nello stato endemico può essere trovata ponendo $dx/dt = 0$ e quindi ottenendo:

$$x = \frac{\beta - \gamma}{\beta}$$

Vediamo quindi che tanto più β è vicino a γ tanto più piccola sarà la frazione di infetti in stato endemico (fino ad arrivare a zero quando $\beta = \gamma$).

6.6.6 Il modello SIRS

Il modello *SIRS*, di cui diamo solo un accenno, complica la situazione. Abbiamo tutti e tre gli stati di susceptible, recovered e ill ma abbiamo che l'immunità acquisita in seguito ad un recover è solo temporanea. Quindi dopo un certo periodo un individuo recovered potrà tornare suscettibile.
Le equazioni differenziali per questo metodo possono essere scritte ma non sono risolvibili analiticamente, richiedono l'impiego di metodi numerici.
Soluzioni a questo modello sono comunque molto variegate e mostrano differenti comportamenti a seconda dei tre parametri di input: ad esempio abbiamo situazioni in cui la malattia persiste in uno stato endemico, casi in cui la ma-

lattia si estingue oppure casi in cui la malattia oscilla fra outbreakes e periodi di remissioni.

6.6.7 Randomness: i modelli stocastici

I modelli che abbiamo appena studiato ricadono nella categoria dei *modelli deterministici*. Tali modelli assumono che in un qualsiasi momento un individuo possa essere suscettibile, infetto oppure recovered. Più precisamente si assume che:

- Solamente gli individui suscettibili possano transire in uno stato di infetto e dopo un certo periodo di infezione si presuppone un periodo di immunizzazione entro il quale l'individuo non può più essere infettato.
- La popolazione è costante.

Abbiamo anche visto che una quantità molto importante è $R_0 = \beta/\gamma$, il basic reproduction number. Quando $R_0 > 1$ l'epidemia prende il sopravvento mentre quando $R_0 < 1$ l'epidemia non si diffonde (quindi di fatto l'epidemia non c'è).

Tutti questi risultati si basano sul fatto che la community sia omogenea e che gli individui si incontrino in maniera uniforme fra loro e questa assunzione è di certo molto forte.
Se ad esempio consideriamo la presenza di una epidemia in piccole comunità come ospediali o scuole sembra abbastanza ragionevole pensare che vi sia una buona dose di casualità ed incertezza nel numero finale di infettati (ad esempio seppur avendo $R_0 > 1$ potremmo avere un solo individuo infettato all'inizio e quindi l'epidemia potrebbe non diffondersi mai).
Per questo i modelli deterministici vengono spesso accompagniati da modelli *stocastici* che introducono alcune variabili di casualità e possibilità di errore che possono essere regolate debitamente sfruttando dati reali.
L'uso di modelli stocastici è maggiormente suggerito per studiare l'estinzione di una epidemia.

6.6.8 Modelli epidemici sulle reti

I modelli visti fin ora assumono una popolazione totalmente connessa ovvero si suppone che ogni individuo possa entrare in contatto con tutti gli altri. Evidentemente questa assunzione è poco realistica, ogni persona avrà il suo gruppo di colleghi, amici o vicini coi quali entrerà in contatto mentre invece non entrerà in contatto con tutti gli altri individui.
Il set di potenziali contatti di un individuo è agilmente rappresentabile tramite una rete la cui forma, capiamo bene, potrà avere notevole impatto sul modo con cui la malattia si diffonderà.
I modelli che prevedono l'impiego di reti funzionano allo stesso modo di quelli

che abbiamo visto in precedenza, ma sfruttano la rete per studiare i contatti fra individui invece che supporre il possibile contatto fra ogni individuo.

Definiamo come *trasmission rate* o *infection rate* la probabilità per unità di tempo che l'infezione venga trasmessa tra due individui (uno suscettibile e l'altro infetto) connessi all'interno della rete. Per analogia coi modelli fully mixed (quelli che abbiamo trattato fin ora) tale probabilità è indicata con la lettera β. Il trasmission rate è proprio della malattia, esistono infatti malattie che si diffondono più facilmente rispetto ad altre, ma è comunque condizionato da parametri relativi alla popolazione e all'ambiente.

Modello SI in versione network

I modelli fully mixed che abbiamo trattato (SI, SIR, SIS) possono essere facilmente generalizzati sfruttando una rete. Prendiamo il caso del semplice modello SI.
Si hanno n individui che sono rappresentati da altrettanti nodi nella rete. Dopodiché, al tempo $t = 0$ una piccola frazione x_0 (eventualmente uno solo) di nodi vengono considerati infettati mentre tutti gli altri sono suscettibili. In seguito si eseguono alcuni round temporali e durante ognuno di questi con una probabilità β ogni nodo infetto propaga la sua infezione ai nodi ad esso collegati.
La simulazione può essere talvolta fuorviante poiché è difficile rappresentare tramite una rete la reale connessione fra individui in una situazione reale, ma i risultati che fornisce il modello quando $t \to \infty$ risultano molto interessanti per studiare situazioni reali. Infatti, quando si guarda la situazione a tempo infinito ogni individuo possibilmente infettabile è infettato (gli infetti rimangono infetti per sempre e tutti i suscettibili collegati ad essi prima o poi diventano infetti se il tempo è infinito). Se ne deduce quindi che, secondo questo modello, un certo nodo non sarà infettato solamente se non esiste un cammino che collega quel nodo con un nodo infetto già a tempo 0.

In altre parole stiamo dicendo che a lungo andare la malattia si diffonderà a partire dagli infetti inziali verso tutti i nodi ad esso collegati. Nel caso più fortunato abbiamo quindi una componente soltanto della rete che viene infettato e poi l'epidemia va a morire poiché non ci sono più altri individui collegati a quella componente. Ma come ben sappiamo la maggior parte delle reti presenta una componente connessa molto ampia e poi altre piccole componenti e questo ci porta ad un intuibile risultato: se il primo nodo infetto si trova nella LC allora l'infezione colpirà la maggior parte della rete, altrimenti avremo una infezione contenuta su una delle piccole componenti (o addirittura nessuna propagazione se l'individuo non aveva contatti).
Se scegliamo l'individuo infetto iniziale con distribuzione uniforme allora la probabilità di outbreak sarà quindi S (la frazione di network componente la LC)

mentre sarà $1 - S$ la probabilità di avere un outbreak contenuto (non si diffonde l'epidemia).

Quanto appena osservato è un comportamento interessante e non osservabile nei modelli fully mixed. In quei modelli infatti il risultato finale dell'epidemia era unicamente determinato dalla scelta del modello e dei suoi parametri mentre invece ora abbiamo una diretta dipendenza fra la struttura della rete e come si svilupperà l'epidemia. Abbiamo inoltre introdotto un elemento stocastico (la scelta dei primi individui infetti) che può portare, a fronte di parametri identici e reti identiche, ad una epidemia che si diffonde oppure che muore.

6.6.9 Proprietà late-time del modello SIR

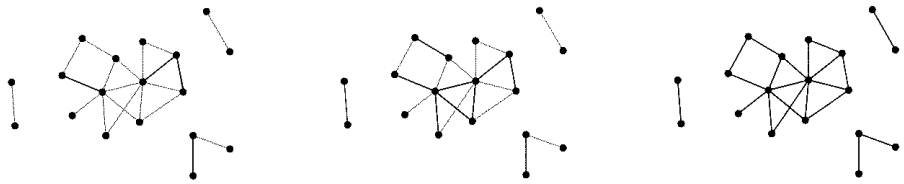
Andiamo ora a studiare anche il caso del modello SIR ma in versione network. In particolare ci interessano le proprietà late-time (a tempo infinito). Ricordiamo che in tale modello gli individui rimangono infetti solamente per un certo periodo di tempo τ e poi si immunizzano. Avevamo poi che la probabilità di trasmissione della malattia è pari a $\phi = 1 - e^{\beta\tau}$ con β il trasmission rate.
Se assumiamo che ogni individuo rimanga infetto per la stessa quantità di tempo (τ, appunto) allora abbiamo che la probabilità di trasmissione è una costante per tutta la rete.
Si ha in definitiva che ogni individuo suscettibile ha una probabilità ϕ di contrarre la malattia da uno dei suoi vicini (quindi se si hanno più vicini la probabilità aumenta).

Percolation

Una delle pratiche che vengono effettuate talvolta in un network è la *percolation*. In sostanza si rimuovono alcuni archi (insieme ai vertici a questi connessi) con l'intento di simulare alcuni fenomeni tipici del mondo reale (pensiamo alla failure di un router). Un altro esempio di fenomeno di percolation nel mondo reale è la vaccinazione o immunizzazione di individui, eventi che evidentemente vanno contro la diffusione della malattia. La vaccinazione di alcuni individui può talvolta portare alla diffusione di una *herd immunity* ovvero anche altri individui diventano immuni senza aver fatto il vaccino. Ancora una volta questo fenomeno è ben descritto tramite azioni di percolazione.

Proviamo ora a prendere il nostro network e applichiamo una *bond percolation* ovvero 'coloriamo' ogni arco con probabilità ϕ. In questo modo stiamo dicendo che gli archi colorati (o occupati) rappresentano quelli attraverso i quali la malattia si diffonderà nel caso in cui uno qualsiasi dei due nodi agli estremi dell'arco sia infetto. Stiamo parlando di una rete di contatti sufficiente a diffondere la malattia ma che non rispetta effettivamente la realtà: se infatti nessuno dei due nodi collegati all'arco è infetto, l'infezione non passerà tramite

quell'arco. L'insieme degli archi è quindi una rete *potenziale* di trasmissione.



(a) $\phi = 0.2$ (b) $\phi = 0.5$ (c) $\phi = 1$

Figura 6.19: Esempi di bond percolation al variare di ϕ.

Tenendo a mente quanto detto, consideriamo la diffusione di una malattia che inizia con un nodo infetto scelto casualmente. Possiamo immediatamente vedere che l'insieme dei vertici che rimarranno infettati è esattamente la serie di vertici connessi a quello iniziale tramite un qualsiasi cammino costituito dai nostri archi colorati. La malattia semplicemente passerà da un vertice all'altro percorrendo gli archi colorati finché tutti i vertici raggiungibilil saranno infettati. Il risultato finale è che la malattia infetterà tutti i membri del percolation cluster (l'insieme dei nodi toccati da archi colorati durante la bond percolation).
Se siamo dunque interessati ad una analisi long-time è sufficiente applicare la bond percolation e calcolare la quantità di nodi nel percolation cluster: tale quantità rappresenta il numero totale di individui infettati durante l'infezione. Quando la probabilità ϕ è bassa notiamo che si creano dei piccoli cluster disconnessi mentre con l'aumento della probabilità di trasmissione man mano i cluster si vanno a fondere creando una grande componente sufficiente da permettere la diffusione della malattia. Quando i cluster iniziano a fondersi abbiamo il cosiddetto *percolation transition*. Tale punto corrisponde precisamente all'epidemic threshold che abbiamo già studiato. Queste interessanti correlazioni fra la percolazione e la diffusione dell'epidemia ci permettono di fare calcoli importanti in merito alla diffusione della malattia.
È comunque importante notare che anche se ϕ si trova sopra all'epidemic threshold è possibile che la mattia non si diffonda. Ad esempio il primo infetto potrebbe non entrare in contatto con abbastanza persone, e questo si verifica nella teoria con probabilità $1 - S$. Chiaramente tale valore sarà tanto più piccolo tanto più siamo sopra l'epidemic threshold.

La percolation è comunque un processo stocastico: due diversi outbreaks che accadano con le stesse condizioni nello stesso network potrebbero percorrere diversi archi inclusi nel percolation cluster e i clusters potrebbero essere di forma diversa. Questo vuol dire che in un caso potrebbe appartenere alla giant component potrebbe non appartenervi nell'altro e quindi portare a risultati di-

versi. La cosa migliore da fare è quella di calcolare dei risultati basandoci su comportamenti medi.

6.7 Game strategies

La game theory è lo studio formale del conflitto e della cooperazione. L'impiego della teoria dei giochi è utile quando decisioni di più agenti (singoli o gruppi) sono interdipendenti fra loro e si vuole cercare di prevedere o semplicemente studiare il loro comportamento.
L'argomento risulta particolarmente interessante poiché rappresenta un altro aspetto dinamico di una rete. Abbiamo infatti trattato l'argomento del contagio e della navigabilità in cui la dinamicità era in un certo senso costretta da leggi esterne mentre invece quando si parla di game strategies si hanno nodi in grado di *ragionare* e dunque prendere decisioni in modo autonomo. Non resta che iniziare.

6.7.1 Esempi introduttivi

Il dilemma del prigioniero

Il primo esempio che andiamo a studiare è il famoso *dilemma del prigioniero*. Ci troviamo nel campo dei giochi con agenti *non-cooperativi* e che quindi sono in conflitto di interessi fra loro.
Abbiamo due presunti criminali, sospettati per aver commesso un certo crimine insieme. I due individui vengono interrogati contemporaneamente ed in stanze separate, dunque non hanno modo di comunicare o di scambiarsi informazioni. Ognuno di essi potrà scegliere se confessare oppure negare di aver commesso il delitto. Rappresentiamo il payoff di questo gioco tramite questa figura:

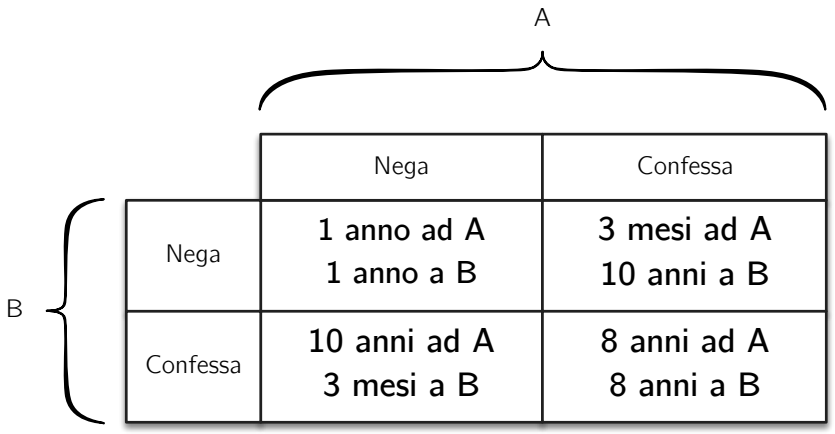


Figura 6.20: Payoff per il dilemma del prigioniero.

La domanda che ci poniamo è: qual è la miglior scelta per un prigioniero? Seppur la combinazione in cui sia A che B negano sembri essere la migliore, la scelta migliore è quella di confessare. Questo poiché qualunque sia la scelta presa dall'altro criminale, il proprio payoff sarà massimo. Questa situazione è detta *equilibrio di Nash* e consiste sostanzialmente nell'intento di un player di massimizzare il suo payoff qualunque sia la scelta dell'avversario. Più precisamente Nash dimostra che, sotto certe condizioni, esiste sempre una situazione di equilibrio, che si ottiene quando ciascun individuo che partecipa a un dato gioco sceglie la sua mossa strategica in modo da massimizzare il suo payoff, sotto la congettura che il comportamento dei rivali non varierà a motivo della sua scelta (vuol dire che anche conoscendo la mossa dell'avversario, il giocatore non farebbe una mossa diversa da quella che ha deciso).
Per sapere se una certa situazione è in equilibrio di Nash è necessario per ogni giocatore chiederci se questo sia interessato a mutare il suo stato (con l'obiettivo di massimizzare il suo payoff). Se nessun giocatore desidera cambiare la sua situazione (poiché non gli conviene) allora abbiamo un equilibrio di Nash. Non si tratta quindi di un equilibrio 'equo', ovvero dove ogni giocatore è 'equamente felice' ma bensì un equilibrio per il quale a nessun giocatore conviene muoversi per migliorare la sua posizione (per quanto tragica possa essere).

Il matching penny

Consideriamo ora un esempio diverso. Abbiamo due giocatori che dispongono di un penny e segretamente lanciano la moneta. Se la partita finisce con entrambe le monete con testa o entrambe le monete con croce, il giocatore A riceve un dollaro dal giocatore B, altrimenti è il giocatore B a ricevere un dollaro dal giocatore A. Il payoff è quindi il seguente:

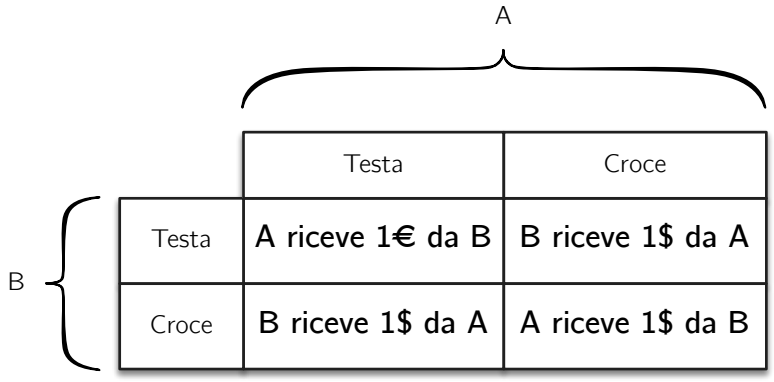


Figura 6.21: Payoff per il gioco matching penny.

Come vediamo questo gioco non dispone di una strategia di equilibrio di Nash cioè non esiste *equilibrio deterministico*: nessuna strategia a priori è migliore

di un altra, è dunque impossibile fornire una *strategia pura*. Esiste invece una scelta più vantaggiosa in termini probabilistici e dunque esiste un *equilibrio stocastico*, ovvero è delineabile una *strategia mista* (una distribuzione di probabilità sull'insieme di strategie possibili che un player può intraprendere).
Ogni strategia pura può essere vista come un caso particolare di strategia mista, che assegna probabilità pari a 1 a quella strategia pura (a tutte le altre strategie pure sarà assegnata probabilità pari a 0).
Nash dimostra che tutti i giochi con giocatori finiti e numero di mosse finito hanno un equilibrio stocastico oppure deterministico.

6.7.2 Giochi su reti

Una volta delineati i concetti di base sui quali si basa la game theory, possiamo andare a parlare dei *giochi su reti*, con lo scopo di andare ad analizzare la rete per capire le proprietà strutturali del problema trattato.
Più precisamente siamo interessati a capire come una topologia di rete possa influenzare le decisioni dei singoli giocatori e quindi anche l'equilibrio definitivo del gioco. Nei giochi su reti abbiamo una serie di giocatori che sono rappresentati da nodi e che possono intraprendere delle strategie (proprio come visto in precedenza). I giocatori possono poi interagire o comunque essere collegati (dipende dal problema) e questo è rappresentato tramite un arco fra due giocatori. Spostarci da una struttura matriciale come quella vista prima verso una struttura a rete permette di studiare casi più complessi e configurazioni più interessanti di player. Come vedremo però all'aumentare dei player avemo anche (tendenzialmente) un aumento molto rapido dello spazio delle soluzioni ovvero delle strategie intraprendibili da ogni giocatore e questo limita in ogni caso i nostri modelli.
I giochi su rete diventano ancora più interessanti quando si parla non di giochi single-shot bensì di giochi che vengono ripetuti iterativamente condizionando man mano la rete e generando nuovi equilibri (tendenzialmente dettati dalla cooperazione degli individui della rete).
A scopo illustrativo parleremo ed esamineremo nel dettaglio tre diversi giochi su reti (più semplici del dilemma del prigioniero che, generalizzato in un ambiente a rete, risulta parecchio ostico).

Il problema della colorazione

Il primo gioco su rete che vediamo è il famoso *problema della colorazione*. Abbiamo diversi nodi (giocatori) che devono scegliere, in un certo lasso temporale, un colore da un set predefinito. L'obiettivo è quello di avere allo scadere del tempo una conformazione dei colori tale per cui non esistano coppie di nodi collegati che abbiano lo stesso colore; in altre parole nessun nodo deve avere lo stesso colore di uno dei suoi adiacenti. Se l'obiettivo è perseguito il nodo guadagna un punto, zero altrimenti.

Ogni player dispone di una conoscenza locale che a seconda delle varianti di questo gioco può essere diversa. Noi considereremo il fatto che un certo nodo conosca il colore deciso da tutti i suoi adiacenti e se esiste un arco che collega fra loro i suoi adiacenti.
Anche qui è possibile cercare un equilibrio di Nash; se ad esempio abbiamo i colori rosso, verde e blu allora potremmo avere questa configurazione:

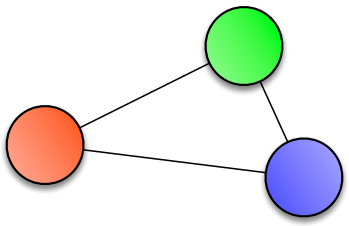


Figura 6.22: Equilibrio di Nash 'buono'.

Ma se invece avessimo solamente i colori verde e rosso allora potremmo ottenere questa configurazione:

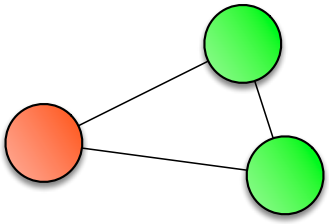


Figura 6.23: Equilibrio di Nash 'cattivo'.

Entrambe le figure rappresentano un equilibrio di Nash, ma nel primo caso abbiamo un equilibrio 'buono' (il payoff di tutti i nodi è 1) mentre nel secondo l'equilibrio è 'cattivo' (due nodi hanno payoff 0).
Già da questi piccoli esempi si capisce che il raggiungimento dell'obiettivo è strettamente dipendente da due fattori: quanti elementi ci sono nell'insieme dei colori e come è configurata la rete.

La domanda a cui vogliamo rispondere è quindi: come può la topologia di una rete modificare l'esito del gioco? La risposta ci giunge da un esperimento effettuato da persone fisiche, che, con l'ausilio di terminali hanno svolto il gioco nel mondo reale. Ciò che si è notato dai dati raccolti è che (paradossalmente) configurazioni semplici come può essere una configurazione a cerchio portano a percentuali di raggiungimento dell'obiettivo molto più scarse rispetto a quelle ottenute in reti complesse come quelle Barabàsi-Albert (con hub). D'altronde la risoluzione del problema con una configurazione a cerchio è molto

semplice se viene effettuata con conoscenza globale (dall'alto) ma è molto più complessa se effettuata con conoscenza locale.

Il problema del consenso

Il problema del consenso è in sostanza il duale di quello appena trattato. L'obiettivo è quello di avere un colore unico su tutta la rete (una decisione che sia condivisa da tutta la rete). Anche per questo problema si è notato che la configurazione della rete risulta un parametro fondamentale.

Il problema del voting

Un esempio ancora più interessante è quello del voting, intrapreso per studiare alcuni fenomeni relativi allo spostamento delle opinioni durante una campagna elettorale. Questi studi sono stati effettuati nel periodo delle elezioni Obama vs Clinton.
Anche qui si deve porre una scelta ma questa volta il meccanismo di payoff è più complesso poiché ogni nodo ha una preferenza intrinseca. Consideriamo ad esempio una configurazione per la quale un nodo che preferisca l'elezione di Obama avrà un payoff di un dollaro se vince Obama e di mezzo dollaro se vince Clinton. Ma potremmo anche avere che un sostenitore di Clinton abbia un payoff di due dollari se vince Clinton e un quarto di dollaro se invece vince Obama.
Questo tipo di impostazione asimmetrica fra le parti è certamente un fattore di influenza, insieme (come sempre) alla conformazione della rete.

6.7.3 Network formation games

Notiamo due fatti apparentemente scorrelati: tutti i giochi di cui abbiamo parlato fin ora trattano il problema della scelta di una strategia data una rete prestabilita ed i modelli per reti che abbiamo studiato sono tutti modelli basati sulla probabilità e che quindi hanno sempre un ingrediente stocastico. Questo secondo fatto è poco realistico: nessuno di noi aggiunge amici su Facebook a casaccio.
Se interpretassimo invece il problema della generazione di una rete come un gioco, allora potremmo ottenere una rete *creata seguendo una strategia*, cosa assai più vicina alla realtà. Questa è l'idea che si cela dietro ai *network formation games* di cui vogliamo fare un accenno in questo paragrafo.
L'idea quindi è quella di studiare le proprietà derivanti da una rete che è stata generata a seguito di una o più 'partite'. Vale sempre il ragionamento già fatto in precedenza: se la rete che vogliamo generare è molto complessa allora questo approccio può risultare poco intelligente (il numero di strategie possibili aumenta esponenzialmente al crescere dei nodi). Consideriamo ora una strategia (molto semplice) di creazione di rete e poi andiamo a vedere se si è

o meno in una situazione di equilibrio di Nash.

La strategia è banale: la creazione di un arco (bidirezionale) ha un costo fisso, C, impostato a 0.99 euro. Il payoff per ogni nodo è quello di appartenere ad una componente connessa il più grande possibile, ovvero ogni nodo facente parte della componente da un payoff 1. Prendiamo subito questo esempio:

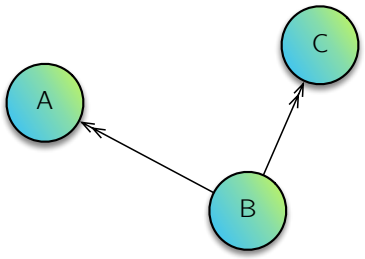


Figura 6.24: Una configurazione di rete per un network formation game.

Giacché gli archi sono bidirezionali rappresentiamo con una doppia freccia chi ha acquistato l'arco.
Abbiamo che B ha comprato due archi (spendendo $0.99 \cdot 2 = 1,98$) ma guadagna 3 poiché la componente connessa di cui fa parte consta di tre nodi (A, B e C). Dunque il payoff totale di B risulta essere $3 - 1.98 = 1.02$. I nodi A e C invece hanno un payoff di 3 (e nessuna spesa). La situazione sembra tutt'altro che d'equilibrio, eppure c'è equilibrio di Nash.
Infatti, nessuno dei player è incentivato a cambiare il suo stato (vendere o comprare nodi) poiché A e C spenderebbero di più avendo la stessa componente e se B vendesse un arco otterrebbe payoff di $2 - 0.99 = 1.01$ che è un centesimo in meno di quanto guadagni ora.

Vediamo invece due casi in cui non c'è equilibrio di Nash (sono riportati a fianco dei nodi i payoff già detratti dell'eventuale spesa effettuata per gli archi):

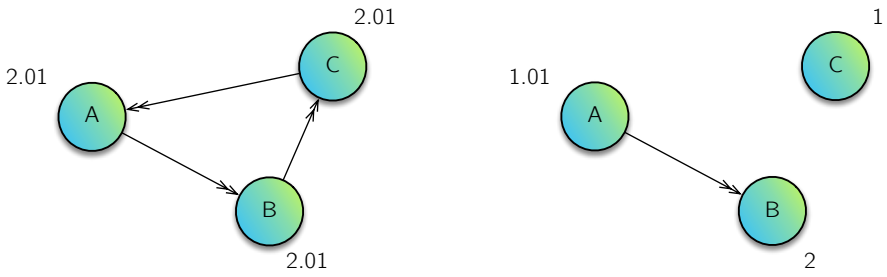


Figura 6.25: Due configurazioni dove non c'è equilibrio di Nash.

Nel primo caso abbiamo che ogni nodo vorrebbe vendere un arco mentre nel secondo caso abbiamo che ogni nodo desiderebbe comprare un arco.

Capitolo 7

Algoritmi sulle reti

Andiamo ora ad analizzare quali algoritmi siano disponibili per trovare gli indici di cui abbiamo parlato fin ora.

7.1 Algoritmi principali

Fra gli algoritmi principali troviamo quelli per il calcolo del grado, della distribuzione del grado, del coefficiente di clustering e dei cammini minimi (che sono indipensabili per calcolare la betweenness, closeness, ecc.).

7.1.1 Grado e distribuzione del grado

Trovare il grado k di un nodo ha costo $O(1)$ se usiamo liste di adiacenza (è sufficiente contare il numero di elementi nella lista).

7.1.2 Distribuzione su ogni nodo

Il calcolo della distribuzione su tutti i nodi è leggermente più complesso. Per ogni grado k si parte inizializzando un $p_k = 0$, dopodiché per ogni nodo i calcoliamo il grado k_i ed aggiorniamo il p_{k_i} incrementandolo di uno. Una volta terminato il calcolo su tutti i nodi (su tutti i loro gradi, quindi) si procede a dividere ogni p_k per n (il numero di nodi). La complessità è evidentemente $O(n)$.

7.1.3 Distribuzione cumulativa

La distribuzione cumulativa è implementabile in due modi diversi. Potremmo usare la tecnica descritta prima per calcolare tutti i p_k, dopodiché osserviamo che:

$$P_k = \sum_{k'=k}^{\infty} p_{k'} = -p_{k-1} + \sum_{k'=k-1}^{\infty} p_{k'} = P_{k-1} + p_{k-1}$$

Questo ci da una traccia algoritmica per calcolare la cumulativa in maniera incrementale poiché:

$$P_k = P_{k-1} + p_{k-1}$$

iniziando da $P_0 = 1$ per definizione. La complessità risulta essere $O(n)$.

Un'altra soluzione è invece:

- Considerare per ogni vertice i la sua k_i (costo $O(n)$).
- Ordinare per ordine decrescente di grado i nodi.
- Si assegna ad ogni nodo i un rank r_i da 1 ad n (seguendo l'ordinamento ovviamente).
- Fare un plot del rank dividendo ogni r_i per n.

La complessità risulta maggiore, $O(n \cdot \log n)$, però tutte le librerie dispongono di un metodo di ordinamento (persino excel).

7.1.4 Coefficiente di clustering

Come sappiamo esiste un coefficiente di clustering locale:

$$C_i = \frac{\text{num di coppie di vertici di i che sono connesse}}{\text{num di coppie di vicini di i}}$$

E poi una definizione globale (detta anche transitività):

$$C = \frac{\text{num di triangoli} \times 3}{\text{num di triple connesse}}$$

Siamo interessati ad un calcolo globale. Possiamo procedere come segue: per quanto riguarda il numeratore, per ogni vertice i possiamo considerare tutti i j, l vicini di i e verifichiamo se esiste un arco nel grafo che colleghi j ed l.
Il numero totale di triple risulta: $\sum_i \frac{1}{2} k_i(k_i - 1)$ per cui il numero totale di confronti da fare è $\sum_i \frac{1}{2} k_i(k_i - 1) = \frac{1}{2} n(\langle k^2 \rangle - \langle k \rangle)$ (si riveda Sezione 5.4.5).
Notiamo quindi una forte correlazione fra distribuzione del grado e complessità dell'algoritmo (come era prevedibile che fosse). Se la distribuzione è a legge di potenza, abbiamo che $\langle k^2 \rangle$ diverge per $\alpha < 3$, il che è terribile in termini di complessità! Mentre gli altri algoritmi che abbiamo visto dipendono da m e quindi indirettamente da $\langle k \rangle = 2m/n$, per il coefficiente di clustering abbiamo anche una dipendenza per $\langle k^2 \rangle$ che è un parametro indipendente.

Cerchiamo di stimare la complessità dell'algoritmo sfruttando quanto detto. Se usiamo liste di adiacenza ci impieghiamo tempo $O(k_i)$ per elencare i vicini di

un vertice i. Dopodiché per ogni l e j appartenenti all'insieme Γ_i (l'insieme dei vicini di i) tali che $j < l$ (li vediamo in maniera ordinata così da non contarli due volte) abbiamo un numero di coppie possibile pari a $\frac{1}{2}k_i(k_i-1)$. Il confronto ci costa dunque $O(k_j)$ oppure $O(k_l)$ (cioè andiamo a cercare l'altro nodo in una delle due liste). Tipicamente avremo un tempo medio che è la media fra i due tempi, quindi $O(k_j + k_l)$. Abbiamo stimato il costo di confronto fra due vicini, ma noi sappiamo bene che vogliamo stimare *tutti i vicini* di i, pertanto abbiamo:

$$\sum_{j,l\in\Gamma_i,\ j<l}(k_j+k_l) = \frac{1}{2}\sum_{j,l\in\Gamma_i,\ j\neq l}(k_j+k_l) = \sum_{j,l\in\Gamma_i,\ j\neq l}(k_j) = (k_i-1)\sum_{j\in\Gamma_i}k_j$$

Quindi il tempo per calcolare il numero di triangoli è proporzionale a:

$$(k_i-1)\sum_{j\in\Gamma_i}k_j = \sum_{ij}A_{ij}(k_i-1)k_j = \sum_{ij}A_{ij}k_ik_j - \sum_j k_j^2$$

Infatti $\sum_i A_{ij} = k_j$. Questa formula ricorda molto quella del calcolo dell'*assortative mixing* (Sezione 3.7.3). Insomma possiamo dire che il tempo per calcoalre il numero di triangoli dipende dal fatto che i vertici *siano correlati o meno per grado*.

Ripartendo dal risultato di prima, in una rete senza correlazioni il tempo per calcolare il numeratore del coefficiente di clustering (cioè il numero di triangoli) è proporzionale a:

$$\sum_{ij}A_{ij}k_ik_j - \sum_j k_j^2 = \frac{1}{2m}\left[\sum_i k_i^2\right]^2 - \sum_j k_j^2 = n\langle k^2\rangle\left[\frac{\langle k^2\rangle}{\langle k\rangle}-1\right]$$

Poiché $2m = \sum_i k_i = n\langle k\rangle$.

Tirando le somme di quanto detto fin ora abbiamo a numeratore un costo:

$$O\left(n\langle k^2\rangle\left[\frac{\langle k^2\rangle}{\langle k\rangle}-1\right]\right)$$

Mentre a denominatore si ha un costo $O(n)$ (il calcolo delle triple è facile, è sufficiente calcolare $\sum_i k_i(k_i-1)$ in una rete gestita come lista di adiacenza).
Evidentemente il costo maggiore è rappresentato dall'elemento $\langle k^2\rangle$ del numeratore.
In una rete reale di n nodi abbiamo un cut-off naturale della distribuzione di grado (non può essere infinita), eventualmente con $k_{max} = n!$. In questi casi, come avevamo già detto, otteniamo:

- $\langle k^2 \rangle = n^{3-\alpha}$
- $\langle k \rangle$ costante (priva di significato).

Pertanto, il tempo per calcolare il coefficiente di clustering in una rete ad invarianza di scala reale è:

$$O\left(n\langle k^2\rangle\left[\frac{\langle k^2\rangle}{\langle k\rangle}-1\right]\right) = O(n \cdot n^{3-\alpha} \cdot n^{3-\alpha}) = O(n^{7-2\alpha})$$

Quandi quando $2 \leq \alpha \leq 3$ avremo una variazione di complessità compresa fra $O(n)$ e $O(n^3)$. Quando $\alpha < 2$ i problemi sono davvero seri!
Migliorare l'algoritmo sarebbe possibile, ma solo tramite l'uso di matrici di adiacenza (che sono però estremamente costose per altri versi). Per questo approcci ibridi risultano la migliore soluzione. In un grafo con $k_{max} = n$ avremo un tempo pari a $O(n^{4-\alpha})$ e quindi complessità decisamente migliori.
Esistono comunque altre proposte in grado di stimare il coefficiente di clustering, la più nota è del 2005, ad opera di Schank e Wagner.

7.1.5 Calcolo dei cammini minimi

È possibile usare ricerche in ampiezza (salvandosi il path durante il calcolo) così da calcolare anche la closeness centrality. Esistono varianti con costo $O(m+n)$.
La betweenness centrality è basata sul calcolo dei cammini minimi. È possibile calcolarli tutti a costo $O(m+n)$ e poi contare quanti ne passano per un vertice ($O(n(m+n))$).
Se i grafi sono pesati possiamo usare l'algoritmo di Dijkstra ($O(n \cdot \log n)$).

7.2 Algoritmo di Ford-Fulkerson

Algoritmo di Ford-Fulkerson è usato per calcolare il *massimo flusso* ed il *taglio minimo*.
L'algoritmo è abbastanza complesso, quindi procediamo per passi. Prima di tutto richiamiamo alcune definizioni.

- Due cammini fra due vertici s e t si dicono $indipendentiperarco$ se non condividono alcun arco.
- Due cammini fra due vertici s e t si dicono $indipendentipervertice$ se non condividono alcun vertice (ad eccezione di s e t).
- La *connettività* è il numero di cammini indipendenti fra due nodi e può essere definita per vertice o per arco (a seconda che consideriamo l'indipendenza per l'uno per l'altro parametro).

- La connettività è equivalente alla dimensione del *taglio minimo*.
- Il taglio minimo è il minimo numero di archi e vertici che devono essere rimossi per disconnettere s da t.

L'algoritmo FF o del massimo flusso (complessità $O\left(\frac{(m+n)m}{n}\right)$) ci fornisce il massimo flusso. Se abbiamo il massimo flusso allora abbiamo la connettività, e quindi la dimensione dell'insieme di taglio minimo.
Esistono due varianti di questo algoritmo: con una possiamo trovare l'insieme di taglio minimo (ovvero tutti gli archi e i nodi che fanno parte del taglio minimo) e invece con l'altra possiamo anche avere i cammini indipendenti.

7.2.1 Idea dell'algoritmo

Consideriamo questo grafo:

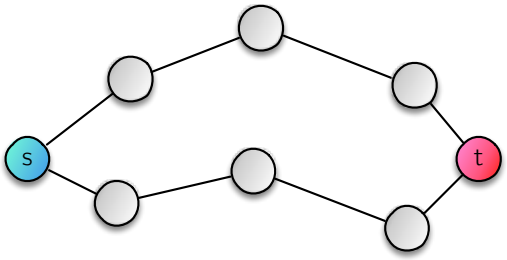


Figura 7.1: Un grafo in cui abbiamo evidenziato s e t.

Per prima cosa, troviamo un cammino da s a t tramite una visita per ampiezza, segnandoci quali archi fanno parte di quel cammino.

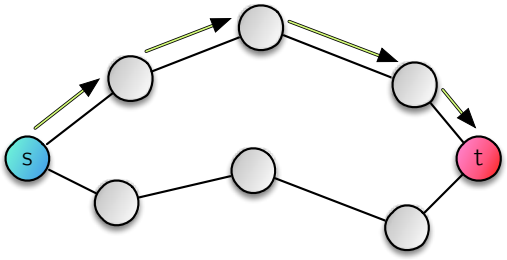


Figura 7.2: Abbiamo trovato un cammino.

Coloriamo quegli archi e li consideriamo usati, dopoiché ripartiamo e cerchiamo un'altro cammino usando altri archi.

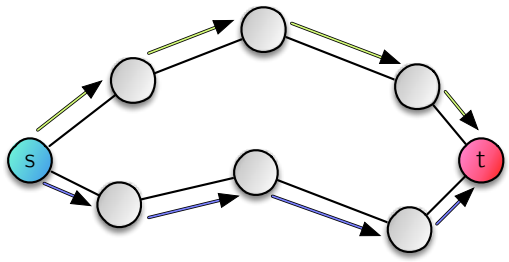


Figura 7.3: Abbiamo trovato un altro cammino.

Proseguiamo in questo modo, finché da s non riusciamo più a raggiungere t. Sembra un metodo abbastanza funzionante... ma con questo grafo:

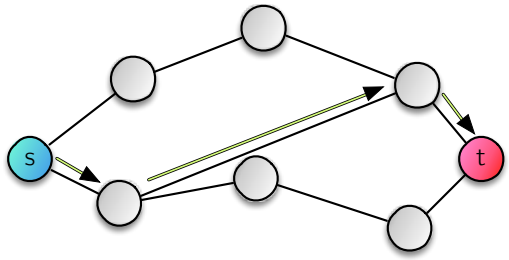


Figura 7.4: Troviamo un solo cammino.

Nonostante vi siano due cammini possibili, ne troviamo solo uno! La soluzione, abbastanza geniale, si basa sul concetto di *grafo residuo*.

7.2.2 Il grafo residuo

Nota: la definizione che daremo è per grafi pesati.
Dobbiamo fare un piccolo salto concettuale: pensiamo alle reti come se fossero traversate da flussi. Ogni arco sarà dunque caratterizzato da una certa quantità x di flusso trasportato, certamente minore della capacità massima u.

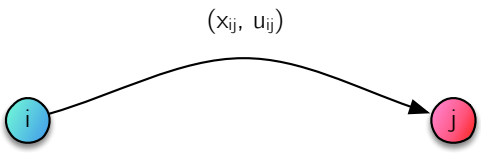


Figura 7.5: Ogni arco è caratterizzato da una x e una u.

È quindi evidente che si possa definire un flusso residuo pari a $u_{ij} - x_{ij}$. Ed è evidente che si possa arrestare un flusso mandandone uno direzione opposta

(da j a i) sempre con intensità x.
Definiamo il grafo residuo come un grafo costruito a partire da quello iniziale, ma dove rispetto ad un dato flusso x, sostituiamo ad ogni arco (i, j) una coppia di archi:

- Un arco (i, j) con capacità residua $r_{ij} = u_{ij} - x_{ij}$.
- Un arco (j, i) con capacità $r_{ji} = x_{ij}$.

Perciò il grafo residuo del piccolo esempio di prima risulta: Cosa stiamo facen-

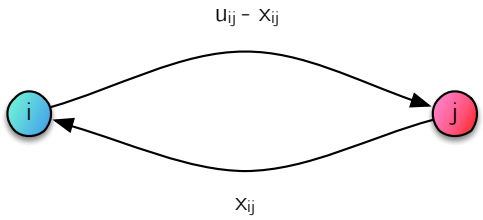


Figura 7.6: Grafo residuo.

do? La capacità residua è ora composta di due componenti, la prima che indica la capacità non utilizzata e la seconda che indica il flusso corrente. Cercare il taglio minimo significa quindi cercare il massimo flusso, *strozzando* tutti gli archi della rete ogni volta che si trovano su un cammino che trasporti qualche flusso.
La rimozione di un arco provoca uno sbilanciamento nell'altra direzione del flusso, perciò, vogliamo togliere l'ultimo arco fra due nodi aggiungiamo invece arco di direzione opposta.

Definiamo:

- *Cammino aumentante* un cammino che porti dalla sorgente al pozzo.
- *Capacità residua* la minima capacità residua fra gli archi nel cammino aumentante.

Non resta che cercare un cammino da s a t nel grafo residuo (quindi un cammino aumentante), una volta trovato ci basterà eliminarlo (cioè eliminarne il flusso relativo) e cercare il prossimo cammino. Quando non si hanno più cammini aumentanti l'algoritmo termina.

Esempio di applicazione

Eliminamo ora tutta la questione dei pesi (e quindi consideriamo tutti gli archi a 1). Ripartiamo dal grafo iniziale e, se non diretto, lo rendiamo diretto.

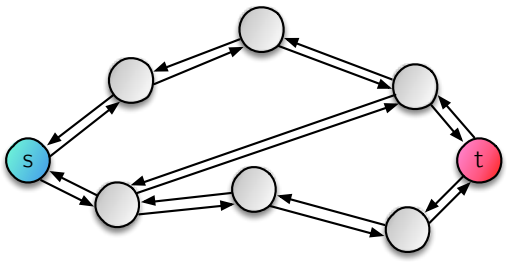


Figura 7.7: Il grafo iniziale.

A questo punto cerchiamo un cammino così come avevamo già fatto prima.

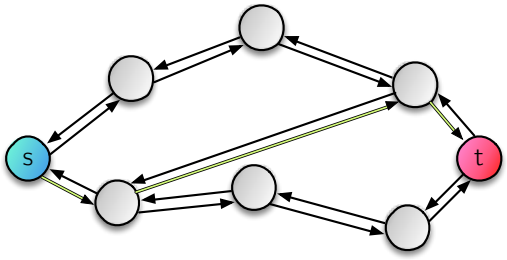


Figura 7.8: Troviamo un primo percorso.

Cancellando il percorso appena trovato (ricordando che l'eliminazione dell'ultimo arco fra due nodi comporta l'inserimento di un arco inverso al posto della rimozione dell'arco stesso, cosa che non accade qui) otteniamo il grafo residuo.

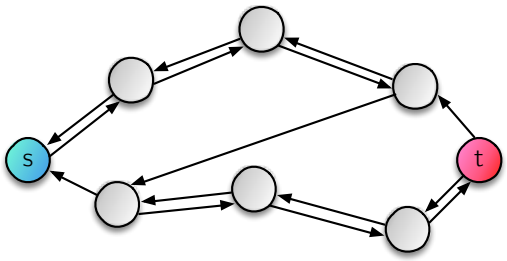


Figura 7.9: Il grafo residuo.

Su questo, cerchiamo un altro cammino.

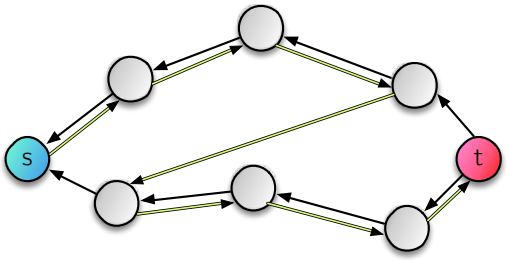


Figura 7.10: Troviamo un altro cammino.

Come fatto prima, cancelliamo tutti gli archi di questo cammino (questa volta dobbiamo ricostituire l'arco inverso per l'arco centrale).

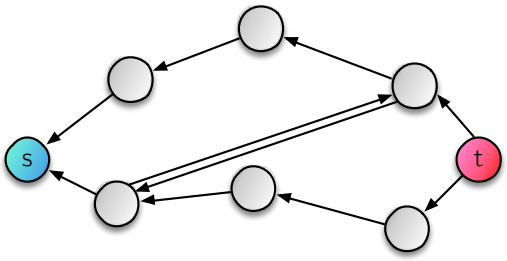


Figura 7.11: Il grafo residuo alla seconda iterazione.

Non siamo più in grado di trovare cammini: il numero totale di cammini risulta essere due.

7.2.3 Teorema del maxflow-mincut

Il massimo flusso fra una data coppia di vertici in una rete è uguale alla somma dei pesi degli archi dell'insieme di taglio minimo che separa i due vertici.

7.2.4 Costo dell'algoritmo

Abbiamo un costo $O(m)$ per costruire il grafo residuo e per aggiornarlo dopo ogni nuovo cammino. Però trovare un cammino equivale ad una visita in ampiezza e quindi ci costa $O(m+n)$, dunque consideriamo questo costo (il maggiore) che deve essere pagato per ogni cammino. Quanti cammini abbiamo? Non possiamo averne più del minimo fra i gradi del nodo s e del nodo t, ovvero $min(k_s, k_t)$. Mediamente questo minimo varrà:

$$\langle min(k_s, k_t) \rangle \leq \langle k \rangle = \frac{m}{n}$$

Perciò moltiplicando il costo per ogni cammino, cioè la visita in ampiezza, $O(m+n)$ per il numero di cammini medi otteniamo $O\left(\frac{(m+n)m}{n}\right)$.

7.2.5 Estensioni dell'algoritmo

L'algoritmo di FF calcola solamente la connettività ma non quali siano i cammini indipendenti. Potremmo essere interessati invece ad averli... allo stesso modo, potremmo essere interessati all'insieme di taglio minimo.

Estensione 1: cammini indipendenti

Prendiamo l'ultimo grafo residuo ottenuto, nel nostro esempio era:

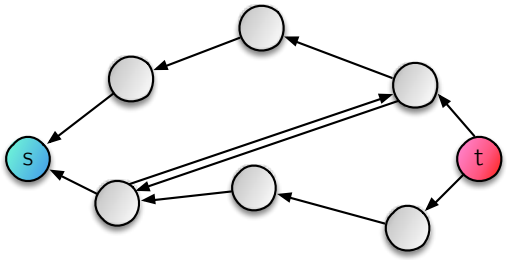


Figura 7.12: L'ultimo grafo residuo.

Rimuoviamo tutte le coppie di archi diretti prodotti dall'algoritmo che uniscono le stesse coppie di vertici ed invertiamo tutti gli archi, ottenendo così i due cammini indipendenti.

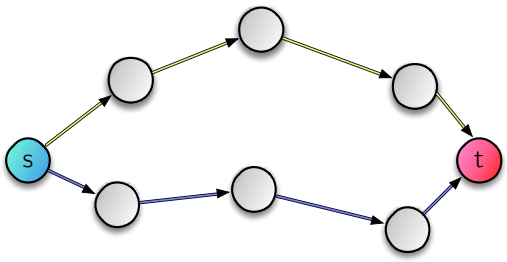


Figura 7.13: Abbiamo i cammini indipendenti.

Estensione 2: insieme di taglio minimo

Se consideriamo l'ultimo grafo minimo, certamente non avremo cammini fra s e t. Creiamo due insiemi:

- V_s: l'insieme di tutti i vertici raggiungibili da s (di cui sicuramente t) non farà parte.
- $V_t = V - V_s$

A questo punto gli archi che collegano V_s con V_t sono l'insieme di taglio minimo. Vediamo un esempio. Abbiamo questo grafo residuo:

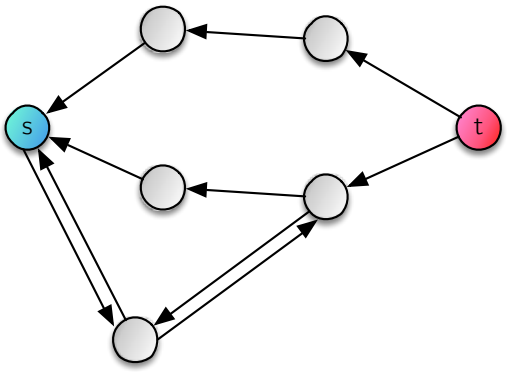


Figura 7.14: L'ultimo grafo residuo.

Deliniamo i due insiemi V_s e V_t secondo la definizione data:

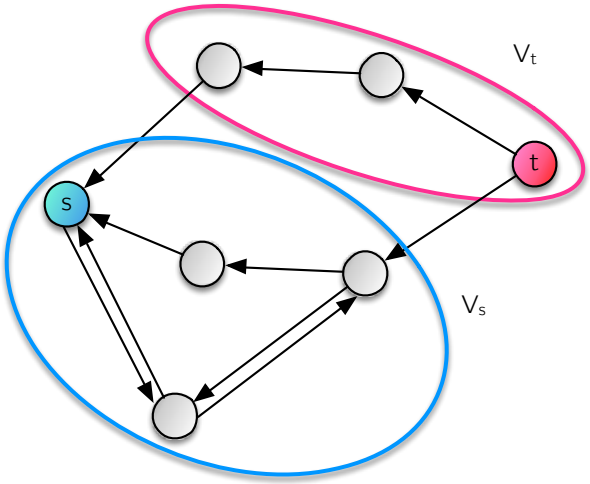


Figura 7.15: Delineiamo V_s e V_t.

A questo punto non resta che prendere quegli archi che sono fra i due insiemi: quello è l'insieme di taglio minimo.

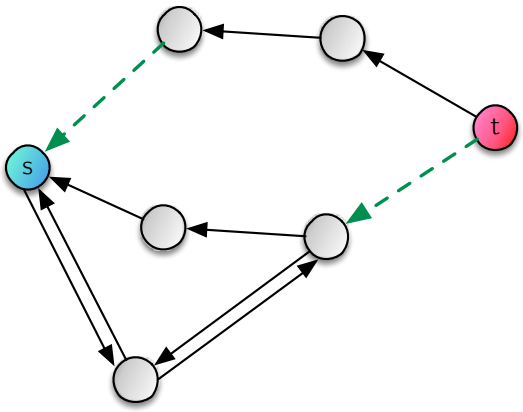


Figura 7.16: I due archi fanno parte dell'insieme di taglio minimo.

Tale insieme è garantito essere minimo, infatti:

- L'assenza di uno degli archi nell'insieme indicherebbe l'assenza di un cammino che usa l'arco corrispondente in senso opposto.
- Se c'è un cammino indipendente che usa quell'arco allora ci deve per forza essere l'arco opposto nel grafo residuo (e infatti lo rinveniamo). Dunque, per sconnettere s e t è necessario rimuovere quell'arco.

Capitolo 8

Community detection

8.1 Introduzione

Abbiamo già accennato alla possibile presenza di comunità all'interno di reti quando avevamo parlato dello studio effettuato da Zachary. In quel caso, poi, un litigio fortuito aveva permesso di verificare la teoria e di dimostrare che effettivamente le due comunità trovate sulla carta esistevano veramente. Si era notato, in quel caso, che le comunità erano riunite intorno a due differenti leader che fungevano da veri e propri magneti per gli altri nodi della rete.

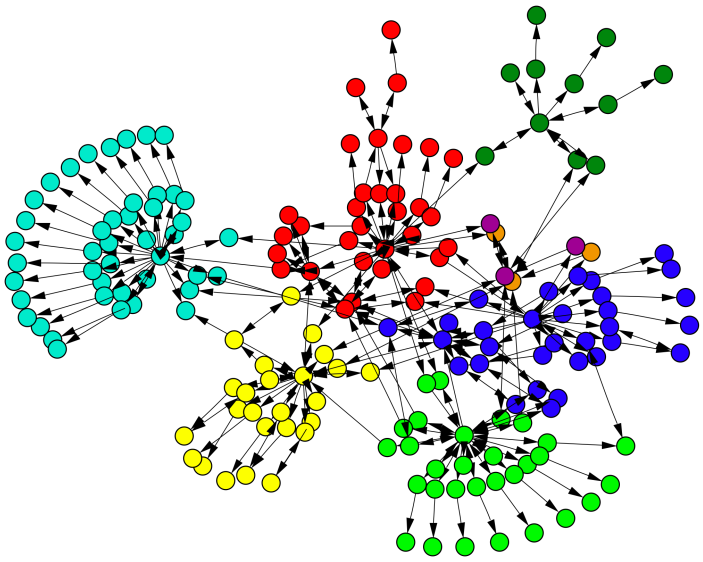


Figura 8.1: Esempio di comunità diverse in una rete di informazioni.

La definizione di *comunità* non è così banale come potrebbe sembrare ed il nome è di per sé un po' improprio. Infatti la definizione propria di comunità (in

ambito sociologico) presuppone che le entità coinvolte facciano parte di una stessa area geografica, si conoscano da vicino e obbediscano a regole comuni. Questi tre fatti non sono invece sempre veri quando parliamo di comunità in una rete sociale.

La struttura a grappoli che contraddistingue una comunità (un sottografo, quindi) non è sempre facile da identificare e non è facilmente evincibile tramite i parametri che abbiamo già studiato (distribuzione di grado, assortativity, coefficiente di clustering). Ma le comunità sono molto importanti e possono darci informazioni preziose in merito ai nodi: per questo il problema della community detection è molto studiato. Tale problema può essere affrontato in modi diversi fornendo più soluzioni (ma risulta complesso capire quale fra queste sia la migliore).

8.1.1 Graph partitioning

Iniziamo il nostro studio introducendo parlando del processo di *parizionamento*. È una operazione che è per molti versi simile a quello che vogliamo fare noi, cioè community detection. Studieremo in tandem questi due processi evidenziandone ove necessario le differenze.
Il parzionamento consta nella suddivisione di un grafo in più parti e la peculiarità di questo approccio è che ci si riferisce a *dimensioni fisse*, cioè sappiamo a priori quanti nodi faranno parte di ogni partizione. Un classico esempio è la suddivisione di n nodi in due partizioni di $\frac{n}{2}$. Le partizioni, in questo caso, hanno intersezione nulla.

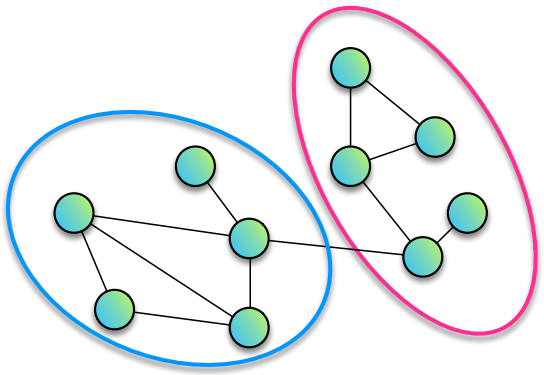


Figura 8.2: Esempio di partizionamento.

Il partizionamento risulta necessario per poter eseguire architetture parallele ad alte prestazioni. Quello che vogliamo ottenere è un partizionamento del

grafo in modo che l'analisi possa essere effettuata in maniera separata per ogni partizione in maniera parallela. Più precisamente: se abbiamo attributi di alcuni nodi che devono essere processati da un processore e quei nodi associati ad un altro processore, avremo una elaborazione inefficiente. Ma se invece cercassimo di assegnare 'per cluster' le operazioni ai processori potremmo ottimizzare lo scambio di informazioni e quindi la computazione. Insomma, una partizione eseguita prima della simulazione del processo dinamico può essere importante ai fini dell'efficenza.

8.1.2 Community detection

Purtroppo il problema della community detection è molto meno chiaro rispetto al partizionamento poiché siamo alla ricerca di una suddivisione *naturale* della rete in gruppi di nodi. È facile dare una definizione intuitiva di comunità: insiemi di nodi collegati da *tanti* archi al loro interno ma che sono collegati da *pochi* archi ad altri insiemi. Ma come definiamo il concetto di 'naturale', di 'tanto' e di 'poco'?
Vediamo una lista di approcci diversi, che approfondiremo in parte in seguito.

Modularity optimization

Un primo approccio, il più popolare, si basa sul seguente ragionamento: la *modularità* (assortative mixing) di una rete è alta quando gli archi collegano nodi simili (ovvero dello stesso gruppo) fra di loro. Sappiamo che nei grafi casuali non esiste modularità, quindi per individuarla è sufficiente calcolare il numero di archi interni allo stesso gruppo di nodi e sottrarvi il numero di archi attesi in una rete equivalente dove i nodi sono collegati fra loro in modo casuale. In questo modo abbiamo in mano un valore che ci può indicare la modularità di una certa zona della rete. Non resta che massimizzare questo valore e trovare così la miglior suddivisione.

Rimozione degli archi

L'idea è semplice: rimuoviamo gli archi più deboli fino a quando la rete non inizia a separarsi in più cluster. La debolezza di un arco è quantificabile in molti modi diversi, ad esempio si può usare la betweenness: ad alta betweenness corrisponde un arco debole poiché molti lo sfruttano per essere collegati a tanti altri nodi, dunque ai due estremi dell'arco si trovano probabilmente due comunità diverse.

Aggregazione gerarchica

Si parte da singoli vertici e man mano li si unisce tra di loro per formare dei gruppi. Si tratta di un metodo detto *agglomerative*. L'unione fra più vertici

viene effettuata sfruttando misure di similarità fra nodi.

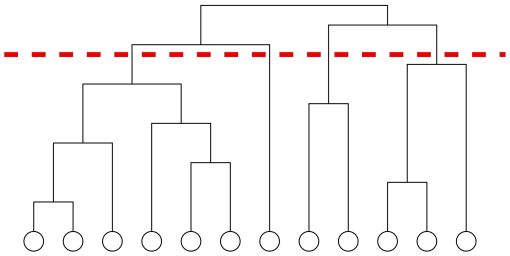


Figura 8.3: Illustrazione di come funziona l'aggregazione gercarchica.

Questa tecnica è molto simile a quella usata nell'ambiente dell'information retrieval per generare i cluster di informazioni.

Metodologie spettrali

La rete viene divisa in due o più componenti sfruttando le proprietà spettrali (relative agli autovalori) della matrice Laplaciana (che sarà l'ultimo argomento che tratteremo). Tali tecniche sono adoperabili sia per effettuare graph partitioning che per fare community detection.

8.1.3 Differenze fra partitioning e community detection

Vi sono principalmente due differenze che contraddistinguono il processo di graph partitioning e quello di community detection:

1. Nel graph partitioning è sempre possibile creare una partizione, mentre nella community detection può non esistere una suddivisione naturale in gruppi (non ci sono comunità).

2. Nel graph partitioning le partizioni hanno intersezione nulla mentre invece le comunità potrebbero avere nodi in comune (overlapping communities).

8.2 Graph partitioning

Adottare un approccio preciso implica una complessità *esponenziale*. Chiaramente questa cosa non ci piace, preferiamo invece ricorrere ad euristiche per migliorare la complessità. Tutti i metodi che tratteremo sono quindi euristici.

8.2.1 Bisezione: algoritmo di Kernighan-Li

Iniziamo studiando il problema di bisezione di un grafo, ovvero vogliamo trovare due gruppi di vertici tali che il numero di archi fra questi sia minimo. Tale numero è detto *cut size*. Il problema che dobbiamo risolvere è simile a quello

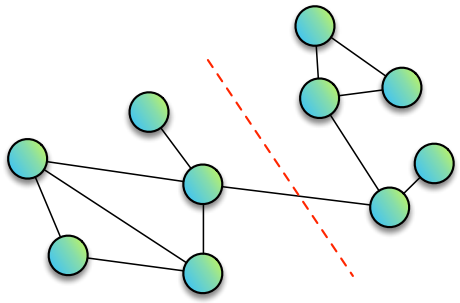


Figura 8.4: Esempio di bisezione.

del massimo flusso ma questa volta non cerchiamo il taglio minimo fra due nodi bensì fra *tutte le possibili bisezioni* della rete.

Soluzione esaustiva

L'approccio più stupido può essere quello esaustivo, ovvero consideriamo tutte le possibili bisezioni del grafo e ne calcoliamo il cut size, dopodiché prendiamo la configurazione con cut size minimo.
Quanto ci costa questo approccio? I numero di possibili bisezioni in una rete di n nodi suddivisa in due insiemi di n_1 ed n_2 nodi (dove ovviamente $n = n_1 + n_2$) è pari a:

$$\frac{n!}{n_1!n_2!}$$

Approssimiamo il fattoriale sfruttando la formula di Stirling secondo la quale:

$$n! \approx \sqrt{2\pi n(\frac{n}{e})^n}$$

Otteniamo quindi:

$$\frac{n!}{n_1!n_2!} \approx \frac{\sqrt{2\pi n(\frac{n}{e})^n}}{\sqrt{2\pi n_1(\frac{n_1}{e})^n_1}\sqrt{2\pi n_2(\frac{n_2}{e})^n_2}} = \frac{n^{n+\frac{1}{2}}}{n_1^{n_1+\frac{1}{2}} n_2^{n_2+\frac{1}{2}}}$$

Per semplicità consideriamo una bisezione in parti uguali, dunque:

$$n_1 = n_2 = \frac{n}{2}$$

A questo punto la nostra formula diventa:

$$\frac{n^{n+\frac{1}{2}}}{\frac{n}{2}^{n+1}} = \frac{2^{n+1}}{\sqrt{n}}$$

Il numeratore cresce più velocemente, dunque la complessità totale risulta essere $O(2^{n+1})$: un brutto esponenziale.

L'algoritmo di Kernighan-Li

Vediamo ora l'algoritmo per la bisezioni di grafi più semplice e più noto, l'algoritmo di *Kernighan-Li* (in breve K-L). L'idea è quella di partire da una bisezione casuale e man mano affinarla riducendo il cut size.
Ad ogni iterazione quindi selezioniamo una coppia di nodi 'promettenti' e li invertiamo, spostandoli da un insieme ad un altro con l'intento di diminuire il cut size. L'immagine a seguire illustra il concetto:

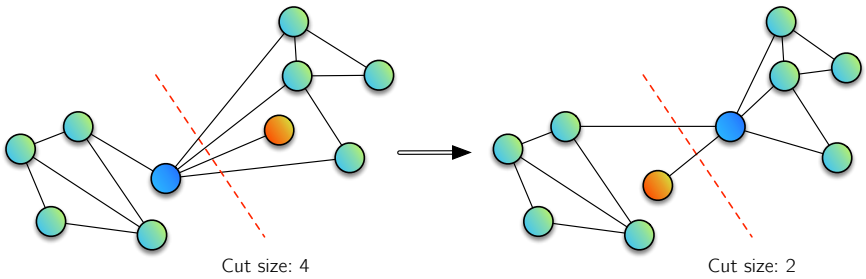


Figura 8.5: L'inversione di due nodi provoca una diminuzione del cut size.

Più formalmente, avendo i due insiemi A e B ed i due nodi $a \in A$ e $b \in B$ da scambiare possiamo definire:

- T, il cut size fra gli insiemi A e B.
- I_a, il numero di archi *interni* fra il nodo a e gli altri vertici in A.
- E_a, il numero di archi *esterni* fra il nodo a e gli altri vertici in B.

Perciò la differenza fra il *costo* esterno e quello interno di a è definibile come:

$$D_a = E_a - I_a$$

Quello che stiamo facendo è considerare un nodo $a \in A$ e vedere quanti archi partono da a andando verso B. Da questi sottraiamo il numero di archi che collegano a a nodi che invece sono in A. Questa quantità D_a è quindi 'quel che guadagneremmo' in termine di archi se spostassimo a dall'insieme A verso l'insieme B. Ma se ripetiamo questo ragionamento per un secondo nodo $b \in B$

allora possiamo calcolare il guadagno ottenuto dallo scambio dei nodi a e b, ovvero:

$$\Delta T = D_a + D_b - 2C_{ab}$$

Abbiamo aggiunto un termine $2C_{ab}$ per rimuovere dal calcolo l'arco fra i due vertici a e b qualora questo esista. Più precisamente, C è la matrice di adiacenza del grafo e quindi C_{ab} è 1 quando l'arco fra a e b esiste. L'arco deve essere rimosso due volte dal conteggio poiché sarà stato conteggiato sia in D_a che in D_b. È naturale che rimuovere tale arco poiché falsa il calcolo di ΔT: se invertissimo a e b quell'arco sarebbe comunque presente e quindi non deve essere considerato.
In pseudocodice:

```
function Kernighan-Lin(''G(V,E)''):
  determina una partizione iniziale dei vertici in A e B
  do
    A1 := A; B1 := B
    calcola D per tutti gli a in A1 e b in B1
      for (i := 1 to |V|/2)
      trova a[i] da A1 e b[i] da B1 tali che
          g[i] = D[a[i]]+D[b[i]]-2*c[a[i]][b[i]] sia max
      sposta a[i] a B1 e b[i] ad A1
      tralascia a[i] e b[i] da ulteriori considerazioni
      aggiorna i valori di D per gli elementi di
          A1 = A1 /{a[i]} and B1 = B1 /{b[i]}
    end for
  trova k che massimizza g_max=g[1] + ... +g[k]
  if (g_max > 0) then
    Scambia a[1],a[2],...,a[k] with b[1],b[2],...,b[k]
  until (g_max <= 0)
  return G(V,E)
```

Listing 8.1: Algoritmo di K-L

Lo scopo dell'algoritmo è quello di trovare la serie di scambi ottimale fra gli elemetni due due gruppi che massimizzi il ΔT.

La complessità di questo algoritmo risulta essere $O(mn^2)$ (che per grafi sparsi con $m \propto n$ sarebbe $O(n^3)$). Pertanto con grafi modesti di 100 o 1000 nodi l'algoritmo K-L può essere una buona soluzione.

8.3 Community Detection

Trattiamo finalmente l'argomento della community detection. Come già detto l'obiettivo è quello di ottenere una suddivisione che sia *naturale*. È evidente che giudicare questa naturalità non sia facile, ma Newman e Girvan (2004) hanno fornito una definizione interessante: giacché i grafi casuali non hanno struttura comunitaria, se confrontiamo un grafo con uno generato casualmente

(ovviamente con stesso numero di nodi ed archi) allora potremmo definire la quantità Q come:

$Q =$ (# di archi *effettivi* nelle comunità) $-$ (# di archi *attesi* nelle comunità)

dove gli archi attesi sono ottenuti dal grafo random e quelli effettivi dal grafo che stiamo studiando.
Evidentemente il nostro obiettivo è massimizzare Q.

8.3.1 Variante di Kernighan-Lin

La prima idea potrebbe essere quella di modificare l'algoritmo K-L che abbiamo studiato in modo che si basi su Q per compiere le sue decisioni. Avremo quindi un algoritmo del genere:

1. Partiamo da una suddivisione casuale.
2. Per ogni nodo v che non è ancora stato spostato calcoliamo quanto cambierebbe il valore di Q se spostassimo v in un altro insieme.
3. Scegliamo fra tutti i nodi esaminati quello che porta ad un Q più alto e marchiamo il nodo v come spostato.
4. Ripetiamo questo processo finché ogni nodo non è stato spostato esattamente una volta.
5. De-marchiamo tutti i nodi segnati come spostati e ripetiamo il processo.

Abbiamo quindi una doppia iterazione: spostiamo ad ogni iterazione più esterna tutti i nodi. Tale ciclo verrà ripetuto finché la modularità non migliora ulteriormente.
La complessità è $O(mn)$: questo algoritmo è uno dei migliori in termini di efficenza.

8.3.2 Massimizzazione spettrale della modularità

Vediamo ora un altro metodo, simile ad un approccio basato sulla matrice Laplaciana e utile per il graph partitioning (non lo vediamo esplicitamente qui).
Partiamo dalla definizione formale di modularità (assortative mixing):

$$Q = \frac{1}{2m} \sum_{ij} \left(A_{ij} - \frac{k_i k_j}{2m} \right) \delta(c_i, c_j)$$

dove ricordiamo che c_i e c_j sono le classi di appartenenza dei vertici i ed j.
Possiamo riscrivere questa formula come:

$$Q = \frac{1}{2m} \sum_{ij} B_{ij} \delta(c_i, c_j)$$

dove evidentemente:

$$B_{ij} = A_{ij} - \frac{k_i k_j}{2m}$$

Questa matrice B che abbiamo appena definito gode di proprietà particolari. Infatti, fissata una riga possiamo effettuare una sommatoria dei suoi j valori e notare:

$$\sum_j B_{ij} = \sum_j A_{ij} - \frac{k_i k_j}{2m} = \sum_j A_{ij} - \frac{k_i}{2m} \sum_j k_j$$

Non abbiamo fatto altro che sfruttare la definizione di B e portare fuori la sommatoria su k_j. Ma la prima sommatoria che abbiamo è la somma dei valori sulle righe, cioè il grado di ogni nodo, quindi k_i; invece la seconda sommatoria ci darà due volte il numero degli archi (sommiamo il grado di ogni nodo, quindi contiamo ogni arco due volte). Possiamo quindi scrivere che:

$$\sum_j B_{ij} = k_i - \frac{k_i}{2m} 2m = 0$$

Notiamo che se lavorassimo su B tenendo fisso j e facendo variare i, la sommatoria porterebbe allo stesso risultato (la derivazione matematica è identica). Abbiamo quindi scoperto che:

$$\sum_j B_{ij} = \sum_i B_{ij} = 0$$

Teniamo bene a mente queste due equivalenze: ci serviranno in seguito.

Troviamo ora una definizione alternativa del delta di Kronecker, più precisamente definiamo:

$$s_i = \begin{cases} +1 & \text{se il nodo } i \text{ appartiene al gruppo 1} \\ -1 & \text{se il nodo } i \text{ appartiene al gruppo 2} \end{cases}$$

Dunque abbiamo che:

$$\delta(c_i, c_j) = \frac{1}{2}(s_i s_j + 1)$$

Ora, la nostra modularità è riscrivibile come:

$$Q = \frac{1}{2m} \sum_{ij} B_{ij} \delta(c_i, c_j) = \frac{1}{4m} \sum_{ij} B_{ij}(s_i s_j + 1)$$

Ora sfruttiamo le due equivalenze di prima per andare ad eliminare il +1: infatti, avevamo che la sommatoria per tutte le i e per tutte le j di B_{ij} ci da 0. Ma da quell'1 usciranno sempre elementi B_{ij} che poi saranno man mano sommati e tutte le componenti sulla i si annulleranno e tutte le componenti sulla j anche.

Si noti che è *la sommatoria ad essere nulla*, non ogni elemento B_{ij}. Infatti il resto del termine, cioè $B_{ij}s_is_j$, rimane. Abbiamo quindi in definitiva:

$$Q = \frac{1}{4m}\sum_{ij} B_{ij}s_is_j$$

che scritto in forma matriciale ci dà l'importante formula:

$$Q = \frac{1}{4m}\vec{s}^T B_{ij}\vec{s}$$

dove $\vec{s}$ è (come prima) il vettore contenente tutti gli s_i e B è una matrice $n \times n$ che contiene gli elementi B_{ij}, anche chiamata *matrice di modularità*.
Ora il nostro problema è diventato trovare il valore di $\vec{s}$ (cioè l'assegnamento in gruppi) che massimizzi Q.
Per massimizzare è necessario differenziare, ma putroppo il nostro vettore $\vec{s}$ può assumere solo valori che siano -1 oppure 1. Cerchiamo dunque un $\vec{s}$ approssimato ma, ovviamente, deve valere il vincolo che:

$$\vec{s}^T\vec{s} = \sum_i s_i^2 = n$$

Cioè stiamo dicendo che comunque la sommatoria dei valori all'interno di $\vec{s}$ deve essere n, cioè dobbiamo assegnare n nodi (il grafo) nella suddivisione in gruppi.
Non resta che derivare per s_i la formula in componenti e imponendo quanto detto sopra sfruttando un moltiplicatore di Lagrange β (in sostanza inseriamo la condizione detta prima, ma in modo che faccia 0 e quindi non intacchi la nostra formula):

$$\frac{\partial}{\partial s_i}\left[\frac{1}{4m}\sum_{jk} B_{jk}s_js_k - \beta\left(n - \sum_j s_j^2\right)\right] = 0$$

Derivando otteniamo:

$$\sum_j B_{ij}s_j = \beta s_i$$

che in notazione matriciale risulta:

$$B\vec{s} = \beta\vec{s}$$

Ma questo non significa altro che $\vec{s}$ è uno dei autovettori della matrice di modularità B!
Per massimizzare Q quindi non resta che scegliere un $\vec{s}$ che sia l'autovettore $\vec{u_1}$ corrispondente all'autovalore massimo della matrice di modularità. Ma non è così semplice... $\vec{s}$ può contenere solamente 1 oppure -1. Dobbiamo quindi

scegliere un $\vec{s}$ che sia il più vicino possibile a $\vec{u_1}$ ma che sia costituito di soli +1 e -1. In formule:

$$\vec{s}^T \vec{u_1} = \sum s_i [\vec{u_1}] i$$

La scelta più opportuna per s_i risulterà quindi:

$$s_i = \begin{cases} +1 & \text{se}[\vec{u_1}]_i > 0 \\ -1 & \text{se}[\vec{u_1}]_i < 0 \end{cases}$$

Se invece $\vec{u_1} = 0$ qualunque valore di s_i è valido (come vediamo dall'ultima formula).

Osservazioni algoritmiche

tutto l'algoritmo si riduce quindi alla ricerca dell'autovettore della matrice di modularità corrisponde all'autovalore maggiore. Questo algoritmo funziona bene in pratica ma purtroppo la matrice di modularità non è mai sparsa e normalmente tutti i suoi elementi sono diversi da zero: lavorare con una laplaciana sarebbe meno costoso (come nell'approccio per il graph partitioning di cui abbiamo accennato sopra).
In termini prestazionali non abbiamo grandi differenze rispetto a quanto ottenuto con la variante di Kernighan-Lin: abbiamo sempre complessità temporale $O(n^2)$ se la rete è sparsa e $O(n^3)$ altrimenti.

I metodi che abbiamo studiato fin ora si occupano solo di fare bisezioni (quindi i gruppi predefiniti erano 2) ma sono generalizzabili per poter far emergere *più gruppi*. I metodi che studieremo in seguito sono già pensati per creare suddivisioni più numerose.

8.3.3 Algoritmo di Girvan-Newman

L'algoritmo di Girvan-Newman è un metodo *divisive* (opposto a quelli agglomerative). Man mano si rimuovono degli archi che connetto i gruppi fino a quando questi rimagono isolati. L'idea è quella già accennata nell'introduzione, ovvero cercare gli archi con più alta betweenness e rimuoverli. Ricordiamo che la edge betweenness di un arco è il numero di cammini minimi che passano attraverso quell'arco arco, quindi stiamo cercando di capire quali archi uniscono sono necessarie per unire gruppi locali del grafo che altrimenti sarebbero separati. Se vi sono più cammini minimi fra stesse coppie di vertici che attraversano quell'arco, allora il contributo di ogni cammino viene suddiviso per il numero totale dei cammini minimi fra quei due vertici presi in considerazione. Il calcolo della betweenness sfrutta algoritmi per la ricerca in ampiezza su grafo, dunque ha complessità $O(mn)$.

L'impianto algoritmico è quindi abbastanza semplice:

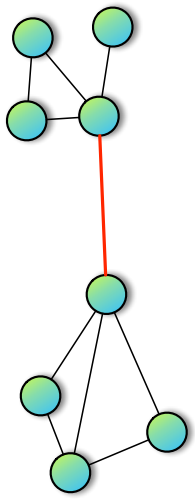


Figura 8.6: Esempio di arco con maggiore betweenness.

1. Si calcola la betweenness di ogni arco.
2. Si rimuove l'arco con più alta betweenness.
3. Ripetere il passo 1.

La complessità è dunque $O(m^2 n)$, cioè $O(n^3)$ con grafo sparso.

Dendrogrammi e terminazione dell'algoritmo

La terminazione dell'algoritmo non è così banale. Vogliamo infatti fermarci quando i gruppi rappresentano il più fedelmente possibile le comunità, ma questo era sostanzialmente il nostro problema iniziale. Che fare?
Introduciamo innanzi tutto la nozione di dendrogramma, che abbiamo già visto quando abbiamo parlato di metodi agglomerative.

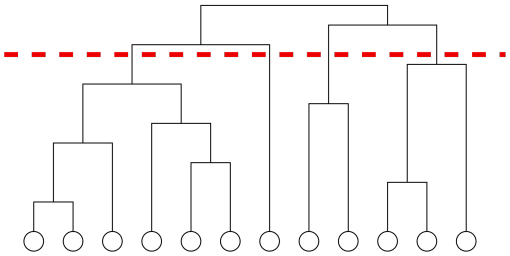


Figura 8.7: Un dendrogramma.

In quell'occasione avevamo detto che la figura si riferiva ad un metodo agglomerative, mentre ora stiamo parlando di metodi divisive. Non c'è nessun

trucco magico, semplicemente dipende da che punto di vista guardiamo il disegno: dall'alto verso il basso andiamo a scindere il grafo in parti e quindi abbiamo un metodo divisive, mentre dal basso verso l'altro agglomeriamo man mano parti di grafo e quindi abbiamo un metodo agglomerative.
L'algoritmo di Girvan-Newman, essendo divisive, può portarci fino ad avere un livello foglia, cioè dove ogni nodo è da solo (è una comunità di per sé).

La terminazione dell'algoritmo quindi consiste in sostanza nello scegliere a che punto del dendrogramma fermarci. Ancora una volta sfruttiamo la modularità: la calcoliamo per ogni decomposizione e poi scegliamo quella con Q massima. La complessità può essere ridotta se si calcola la edge betweenness solo per coppie di vertici scelti a caso piuttosto che considerare tutti gli archi (ancora una volta un approccio euristico).

Variante di Radicchi

È possibile implementare una variante meno costosa dell'algoritmo dove invece di usare la betweenness si rimuovono gli archi che appartengono a *cicli corti*. L'idea è quella che tendenzialmente un arco con alta betweenness appartiene a cicli che non sono lunghi. Possiamo dare una dimostrazione intuitiva per assurdo di questo fatto: diciamo che vi sono nodi vicini fra loro ma di gruppi diversi e che sono connessi direttamente (formano un ciclo), questo implicherebbe il fatto che fra di loro i gruppi sono molto ben connessi (poiché due nodi di gruppi diversi sono invece molto vicini) e quindi in realtà non sono veramente gruppi distinti.

Sperimentalmente si è visto che i risultati migliori si ottengono cercando cicli di 3-4 vertici. Dal punto di vista computazionale abbiamo un costo $O(n)$ per il calcolo dei cicli corti (è un calcolo locale) e quindi un costo totale di $O(n^2)$ nel caso peggiore.
La variante di Radicchi è consigliabile in quelle situazioni in cui ci sono molti short loop (le reti sociali ad esempio) mentre invece con altre reti con pochi cicli corti l'algoritmo fa fatica a discriminare bene fra archi interni ai gruppi ed esterni ai gruppi.

8.3.4 Clustering gerarchico

Il clustering gerarchico è un metodo agglomerative, che quindi è sempre rappresentabile con un dendrogramma ma questa volta ci muoviamo dal basso verso l'alto aggregando man mano i nodi sfruttando una nozione di *similarità*. I gruppi di nodi così ottenuti vengono detti cluster (e ovviamente siamo interessati ad ottenere cluster che siano comunità). Risulta quindi necessario definire il concetto di similarità.

Misure di similarità

Abbiamo già studiato diverse misure di similarità. Ricordiamo:

- Structurale equivalence dissimilarity: $d_{ij} = \sqrt{\sum_{k \neq i,j} (A_{ik} - A_{jk})^2}$
- Neighborhood's overlap $\omega_{ij} = \frac{| \Gamma(i) \cap \Gamma(j) |}{| \Gamma(i) \cup \Gamma(j) |}$
- Pearson correlation coefficent $C_{ij} = \frac{\sum_k (A_{ik} - \mu_i)(A_{jk} - \mu_j)}{n \sigma_i \sigma_j}$ dove:
  - $\diamond$ $\mu_i = \frac{\sum_j A_{ij}}{n}$
  - $\diamond$ $\sigma_i = \sqrt{\frac{\sum_j (A_{ij} - \mu_i)^2}{n}}$

Similarità fra gruppi

Grazie alle misure di cui sopra abbiamo la possibilità di fare il primo passo dell'algoritmo, dove coppie di vertici simili vengono agglomerate formando i primi cluster di dimensioni 2. Ma come definiamo la *similarità fra gruppi* (per i passi successivi)? Non scendiamo nei dettagli, riportiamo solamente i nomi di tre tecniche:

- Single-linkage clustering.
- Complete-linkage clustering.
- Average-linkage clustering.

8.3.5 Approcci greedy

Quando sfruttiamo un approccio greedy ci riferiamo all'uso di informazioni locali e non globali. Il metodo di Newman con variante di Clauset ha complessità $O(n \log^2 n)$ per reti sparse mentre il metodo di Louvain (che studiamo) ha complessità $O(m)$. I risultati in merito alla complessità sono quindi ottimi ma purtroppo il risultato finale è facilmente influenzato dall'ordine in cui si studiano i vertici e spesso si rinvengono partizioni sbilanciate su grafi grandi.

Metodo di Louvain

Il metodo si suddivide in due passi.
Prima di tutto effettuiamo un pre-passo di modularizzazione, dopodiché procediamo con altri due passi:

1. Le comunità vengono collassate in singoli nodi, con un self-loop che ha peso corrispondente al numero di archi facenti parte la comunità (archi bidirezionali contano doppio).

2. Dopodiché si procede con l'unione dei nodi del nuovo grafo (comunità) ottenuto dal passo 1, scegliendo di unire quei nodi che hanno molti archi fra loro.

Vediamo un esempio di esecuzione.

Abbiamo il passo iniziale di modularizzazione:

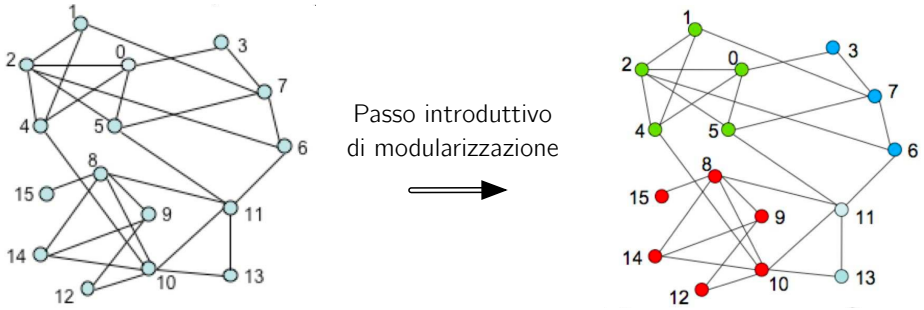


Figura 8.8: Passo introdutivo dell'algoritmo di Louvain.

Procediamo dunque con il primo passo, aggregando le comunità trovate:

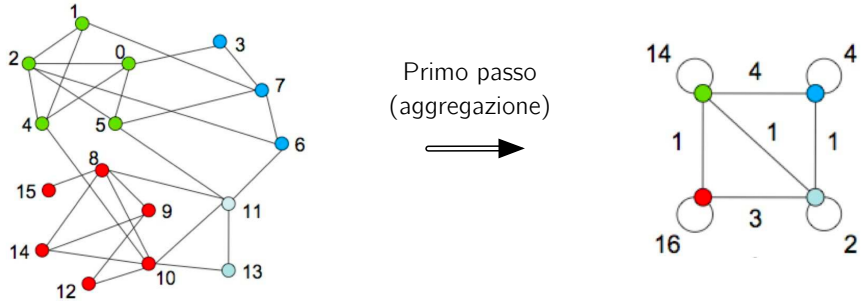


Figura 8.9: Passo di aggregazione.

Non resta che continuare ad aggregare, scegliendo di unire le comunità che hanno più archi che le collegano:

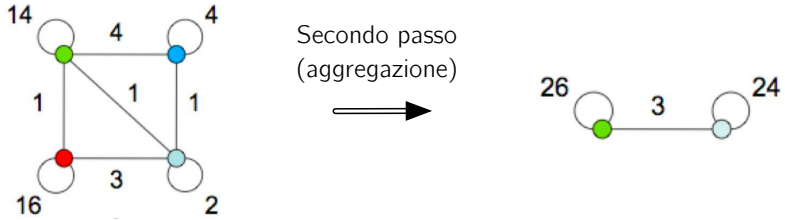


Figura 8.10: Un altro passo di aggregazione.

Come già fatto per l'algoritmo di Girvan-Newman la terminazione si baserà sulla modularità: ad ogni passo calcoliamo Q e non appena questo inizia a decrescere ci fermiamo.

8.3.6 InfoMap

Partiamo da questo ragionamento: i random walk spendono molto tempo all'interno di una comunità prima di uscire, poiché vi sono molti più archi che portano all'interno di una comunità che archi che portano verso altre comunità. La seconda idea è quella di comprimere la descrizione del flusso di informazione contenute in una rete.
Vogliamo quindi fare qualcosa del genere:

1. Descriviamo la traettoria di un random walk in una rete, in modo da dare nomi unici ai nodi. I nomi che diamo sono codici di Huffman (codici compressi, quindi).

2. Dopodiché si iniziano ad assegnare nomi (sempre codificati) a gruppi di nodi, sfruttando l'idea che abbiamo avuto prima.

Alla fine riporteremo solamente i nomi dei moduli che abbiamo ottenuto. Vediamo un esempio.
Prima di tutto individiamo i random walk (primo passo) e poi diamo un nome ai nodi (sfruttando i walk che abbiamo calcolato prima):

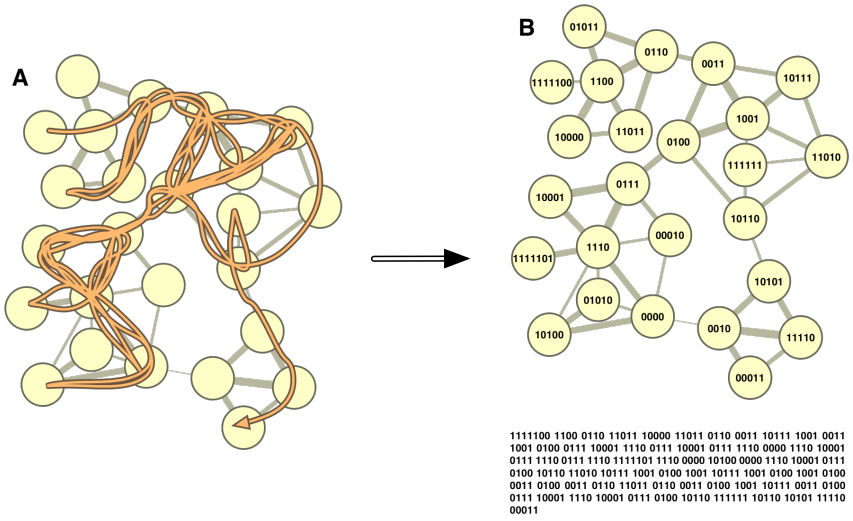


Figura 8.11: Individuiamo i random walk e diamo un codice ai nodi.

Procediamo dunque iniziando a dare nomi alle comunità osservando i prefessi dei vari nodi (a prefissi uguali corrispondono nodi che nel cammino erano succedanei), man mano affinando le comunità.

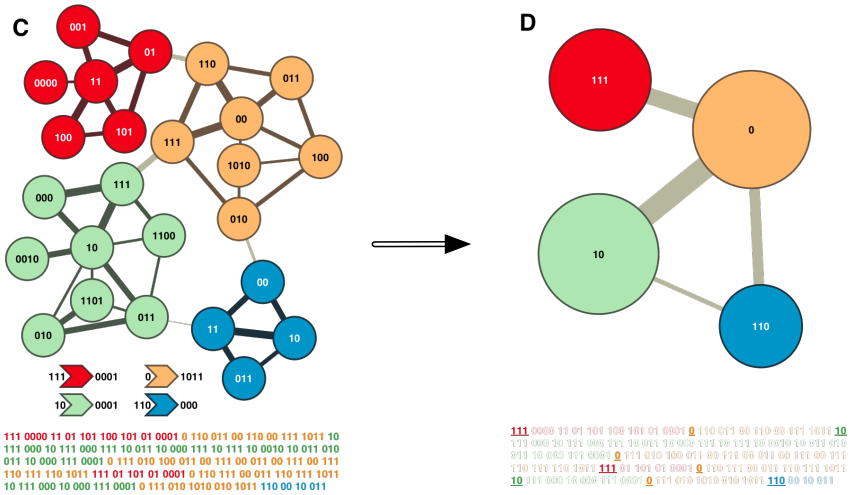


Figura 8.12: Man mano affiniamo le comunità.

L'algoritmo può essere migliorato sfruttando politiche gready.

8.3.7 Resoconto sui metodi

Riportiamo una tabella di Lancichinetti e Fortunato (2009) che riassume le complessità metodi di cui abbiamo parlato.

Author	Label	Order
Girvan & Newman	GN	$O(nm^2)$
Clauset et al.	Clauset et al.	$O(n \log^2 n)$
Blondel et al.	Blondel et al.	$O(m)$
Guimerà et al.	Sim. Ann.	parameter dependent
Radicchi et al.	Radicchi et al.	$O(m^4/n^2)$
Palla et al.	Cfinder	$O(\exp(n))$
Van Dongen	MCL	$O(nk^2)$, $k < n$ parameter
Rosvall & Bergstrom	Infomod	parameter dependent
Rosvall & Bergstrom	Infomap	$O(m)$
Donetti & Muñoz	DM	$O(n^3)$
Newman & Leicht	EM	parameter dependent
Ronhovde & Nussinov	RN	$O(n^\beta)$, $\beta \sim 1$

Figura 8.13: Tabella riassuntiva dei metodi per community detection.

Come vediamo InfoMap e Louvain sono i metodi più efficienti.

8.4 Overlapping communities

Tutti i metodi che abbiamo studiato non considerano la possibilità che ci siano comunità in sovrapposizione, ovvero un nodo possa appartenere a due comunità diverse. Ma questa configurazione è possibile nella realtà dunque

esistono metodi per lo studio di questi fenomeni.

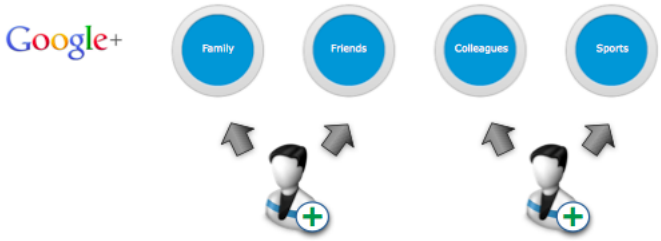


Figura 8.14: In Google+ si possono avere comunità sovrapposte.

Le applicazioni derivabili da questo tipo di configurazioni sono moltissime e si muovono verso i grandi campi delle *tassonomie dinamiche* e dei *sistemi di reccomandation*.
Comunità distinte senza overlap praticamente *non esistono* in reti di grandi dimensioni, anzi, le comunità si sovrappongono in modo pervasivo rendendo impossibile il partizionamento della rete senza sezionare le comunità stesse.
Uno dei metodi più noti per studiare le comunità sovrapposte è il *clique percolation* (sfruttato poi da altri metodi, come vederemo in seguito).

8.4.1 CFinder

Il primo algoritmo che accenniamo, basato sulla clique percolation, è chiamato *CFinder*. L'algoritmo è il seguente:

1. Si identificano delle k-cliques.
2. Coppie di k-clique sono definite adiacenti se condividono $k-1$ nodi.
3. Si formano comunità sovrapposte componendo l'unione massimale di k-clique che possono essere raggiunte l'una dall'altra attraverso k-clique adiacenti.

Vediamo un insieme di comunità sovrapposte costruite a partire da 4-clique:

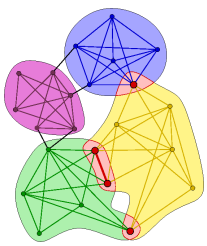


Figura 8.15: Esempio di comunità ottenute dal CFinder.

8.4.2 Altri metodi

Altri metodi che possiamo usare sono:

- *Greedy Clique Expansion (GCE)*: identifica distinte clique come 'semi' ed espande i semi tramite ottimizzazione greedy di una funzione di fitness locale.

- *MOSES*: approccio scalabile per identificare comunità con altro grado di sovrapposizione. Seleziona un arco casualmente ed espande la comunità attorno a quell'arco in modalità greedy cercando di ottimizzare la funzione obbiettivo, dopodiché rimuove le comunità di 'bassa qualità' e termina affinando le comunità riassegnando individualmente i nodi.

- *Link Communities*: come MOSES questo metodo si concentra su gruppi di archi piuttosto che di vertici. È ottimo per individuare comunità che si presentano in modo pervasivo in una rete con caratteristiche *multicale*, cioè con struttura gerarchica (comunità, sottocomunità, sotto-sottocomunità, ecc.).

Vediamo un esempio dell'ultimo metodo:

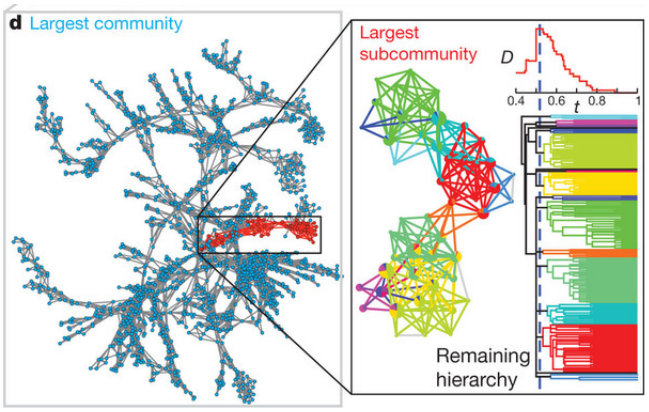


Figura 8.16: Esempio d'uso dell'algoritmo 'link communities'.

8.5 Comunità in evoluzione

Terminiamo le nostre riflessioni in merito alle comunità ricordando che molte delle analisi che vogliamo fare sono in realtà su evoluzioni temporali di una rete.
L'idea è quindi quella di prendere diverse foto istantee della rete, e poi studiare il variare delle comunità nel tempo. Possiamo studiare le comunità in termini

di *nascita e morte* (comunità che prima non c'erano compaiono e comunità già esistenti muoiono) oppure possiamo studiare comunità in *espansione e contrazione* (il numero di nodi nelle comunità aumenta o diminuisce) oppure ancora possiamo studiare comunità nei termini di *fusione e separazione* (comunità che si uniscono o che vengono generate dalla scissione di altre comunità). Riassumiamo tutto in questa figura:

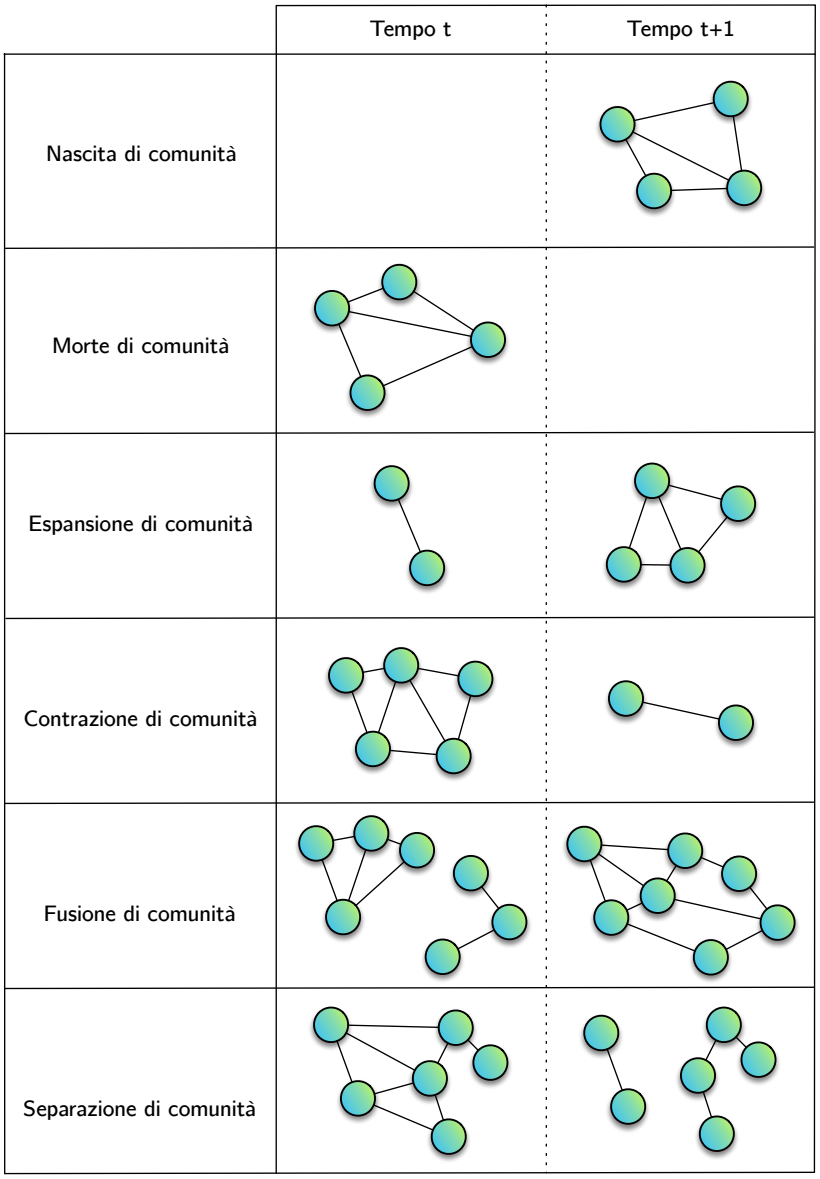


Figura 8.17: Tabella riassuntiva delle tipologie di cicli di vita.

Capitolo 9

Il grafo Laplaciano

Abbiamo più volte accennato di come la matrice Laplaciana sia importante e fondamentale per molti aspetti diversi. Da questa matrice si può ottenere un grafo, detto *grafo Laplaciano*, che come scopriremo è uno strumento notevole che ci può dare grosse informazioni sul nostro grafo.

9.1 La matrice Laplaciana

Nota: consideriamo di grafi senza self-loop.
Iniziamo ricordando la definizione di matrice Laplaciana:

$$L = D - A$$

Dove A è la matrice di adiacenza con elementi a_{ij} è così definiti:

$$a_{ij} = \begin{cases} 1 & \text{se c'è un arco dal nodo } i \text{ al nodo } j \\ 0 & \text{altrimenti} \end{cases}$$

e vi si sottrae la matrice D, che è costituita in questo modo:

$$D = \begin{pmatrix} k_1 & 0 & 0 & \cdots \\ 0 & k_2 & 0 & \cdots \\ 0 & 0 & k_3 & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix}$$

dove ogni k_i è il peso del nodo i, che in termini della matrice di adiacenza, come sappiamo, significa $k_i = \sum_j a_{ij}$.
Ma giacché non vi sono self loop, la matrice di adiacenza avrà tutti 0 sulla diagonale e quindi, per come è costruita, la somma degli elementi sulla riga

della matrice Laplaciana sarà sempre 0. Quindi L sarà in questa forma:

$$L = \begin{pmatrix} k_1 & -a_{12} & -a_{13} & \cdots \\ -a_{21} & k_2 & -a_{23} & \cdots \\ -a_{31} & -a_{32} & k_3 & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix}$$

L'analisi spettrale (relativa agli autovalori) di questa matrice ci dà informazioni utili in merito a:

- Diffusione
- Connettività del grafo
- Partizionamento (è una derivazione della connettività, non ne parleremo esplicitamente).

9.2 Il processo di diffusione e il grafo laplaciano

La *diffusione* è, fra le altre cose, il processo tramite il quale un gas si muove da una regione ad alta densità verso una regione di densità minore, guidato dalla pressione parziale delle due regioni. È possibile studiare processi di diffusione anche sui network, con l'intento di studiare la diffusione, ad esempio, di una idea o di una malattia.

Supponiamo di avere una qualche forma di sostanza distribuita su un network e immaginiamo che che il nodo i-esimo ne disponga di ψ_i. Supponiamo poi che la sostanza si muova attraverso gli archi, fluendo da un vertice j ad un suo adiacente i con andamento $C(\psi_j - \psi_i)$ dove C è una certa costante detta *costante di fluidità*. Insomma in un lasso di tempo la quantità di liquido che è fluito da j ad i è $C(\psi_j - \psi_i)dt$. Quindi l'andamento con il quale ψ_i cambia è dato dall'equazione:

$$\frac{d\psi_i}{dt} = C \sum_j A_{ij}(\psi_j - \psi_i)$$

La definizione che abbiamo appena scritto è abbastanza evidente, la quantità di sostanza in arrivo a i è quella di tutti gli archi adiacenti (ogni ψ_j quindi) meno quella già presente ψ_i. La presenza della matrice di adiacenza ci garantisce che vengano conteggiati i flussi solo su gli archi che effettivamente esistono sulla rete.
Concentriamoci sul caso di un grafo indiretto (ma l'equazione appena trovata vale anche per i grafi diretti) e semplice (no self-loop e un arco è collegato a due vertici soltanto). Lavoriamo sull'equazione di prima dividendola in due parti:

$$\frac{d\psi_i}{dt} = C \sum_j A_{ij}\psi_j - C \sum_j A_{ij}\psi_i$$

Dal secondo termine possiamo però intravedere la definizione del grado di i, $k_i = \sum_j A_{ij}$ e quindi abbiamo:

$$\frac{d\psi_i}{dt} = C \sum_j A_{ij}\psi_j - Ck_i\psi_i$$

Ragionamo ora sul fatto che ψ_i quando $j = i$ è ovviamente ψ_j. Perciò potremmo includere lo ψ_i nella sommatoria e dunque scrivere:

$$\frac{d\psi_i}{dt} = C \sum_j (A_{ij} - \delta_{ij}k_i)\psi_j$$

Il delta di Kronecker ci permette di 'attivare' il k_i solamente quando $i = j$ poiché in quel caso $\delta_{ij} = 1$ (altrimenti vale zero). Osservando attentamente l'equazione ritroviamo alcuni vecchi amici (passiamo in scrittura matriciale):

$$\frac{d\psi_i}{dt} = C(A - D)\vec{\psi}$$

Dove A è la matrice d'adiacenza, D è la matrice diagonale di cui abbiamo parlato prima e $\vec{\psi}$ è il vettore dove ogni elemento è psi_i. Riprendendo quindi la definizione di matrice Laplaciana:

$$L = D - A$$

possiamo scrivere che:

$$\frac{d\psi_i}{dt} + CL\vec{\psi} = 0 \tag{9.1}$$

Questa equazione è simile a quella che descrive la diffusione di un gas, ma invece che il l'operatore Laplaciano ∇^2 appare la matrice L. È per questo che L è anche detto *grafo Laplaciano* e ci da molte altre informazioni oltre a quelle sulla diffusione che abbiamo appena scoperto. Come abbiamo già accennato, gli elementi di L sono:

$$L_{ij} = \begin{cases} k_i & \text{se } i = j \\ -1 & \text{se } i \neq j \text{ e c'è un arco dal nodo } i \text{ al nodo } j \\ 0 & \text{altrimenti} \end{cases}$$

Una definizione alternativa quindi è:

$$L_{ij} = \delta_{ij}k_i - A_{ij}$$

Andiamo ora a risolvere l'equazione Eq. 9.1 scrivendo il vettore ψ come combinazione lineare degli autovettori v_i della matrice laplaciana, cioè stiamo dicendo che:

$$\psi(t) = \sum_i a_i(t)v_i$$

Con un certo coefficiente a_i che varia nel tempo (ci serve per scrivere la scomposizione lineare). Usando questa scrittura nell'Eq. 9.1 e sfruttando la definizione di autovalore per cui $Lv_i = \lambda_i v_i$ dove λ_i è l'autovalore associato all'autovettore v_i, otteniamo:

$$\sum_i \left(\frac{da_i}{dt} + C\lambda_i a_i(t) \right) v_i = 0$$

Ma poiché la matrice L è simmetrica, i suoi autovalori sono ortogonali e quindi facendo il prodotto scalare di questa equazione con qualsiasi autovettore v_j otteniamo:

$$\frac{da_i}{dt} + C\lambda_i a_i(t) v_i = 0$$

per ogni i. Andando quindi a risolvere l'equazione differenziale otteniamo l'importante risultato:

$$a_i(t) = a_i(0) \cdot e^{-C\lambda_i t}$$

Ovvero, possiamo studiare la variazione di informazione per un nodo i tramite il coefficiente a_i (ricordiamo che derivava dal nostro psi_i) che varia nel tempo seguendo la legge che abbiamo appena trovato. La cosa importante da notare è che la relazione è semplice, ci basta sapere a quale autovalore λ_i è associato il nodo e poi calcolare $a_i(t)$ per un tempo t qualsiasi a nostro piacimento (dobbiamo anche conoscere la quantità iniziale di informazione $a_i(0)$). Una volta calcolato $a_i(t)$ sarà sufficiente moltiplicarlo per l'autovettore associato al nodo e otteniamo la quantità di informazione al tempo t cercato.

9.3 Gli autovettori del grafo Laplaciano

L'abbiamo già detto molte volte: spesso ci si ritrova a lavorare con gli autovettori della matrice L (proprio come ci è appena capitato). Abbiamo già notato che gli autovettori di L sono ortogonali ma in realtà ci sono altre informazioni che possiamo ottenere: ad esempio, come stiamo per dimostrare, gli autovalori sono sempre non-negativi.

Consideriamo ora una rete con archi non direzionati con n nodi ed m archi e decidiamo in modo arbitrario che ogni arco ha un estremo che chiamiamo end1 e un'altro che chiamiamo end2. Non ci importa di quale estremo si sta parlando, purché siano etichettati in maniera differente. Poi definiamo una matrice B $m \times n$ con la quale vogliamo rappresentare gli archi del grafo, quindi con elementi definiti come segue:

$$b_{ij} = \begin{cases} +1 & \text{se end1 dell'arco } i\text{-esimo è collegata al nodo } j \\ -1 & \text{se end1 dell'arco } i\text{-esimo è collegata al nodo } j \\ 0 & \text{altrimenti} \end{cases}$$

In questo modo ogni riga della matrice B avrà esattamente un +1 ed un -1 (poiché ogni riga rappresenta un arco, ed è collegato alle due estremità end1 ed end2 con due e due soltanto nodi). La matrice B è chiamata *matrice di incidenza*.
Consideriamo ora la somma:

$$\sum_k B_{ki} B_{kj}$$

Quando $i \neq j$ allora l'unico termine non zero della somma sarà quello per cui né B_{ki} né B_{kj} sono zero (ovviamente, essendo moltiplicati fra loro nessuno deve essere 0) e questo accade solamente quando il nodo i è collegato al nodo j (e il risultato della somma sarà -1 poiché la moltiplicazione ottenuta sarà $+1 \cdot -1$ oppure $-1 \cdot +1$).
Quando invece $i = j$ allora la somma sarà in realtà $\sum_k B_{ki}^2$ che ci darà 1 per ogni vertice collegato ad i (percorriamo tutte le righe tenendoci fissi sulla colonna [del nodo] i, e abbiamo valore +1 o -1 se un qualche arco è collegato con end1 o end2 a quel nodo i; ma qualsiasi sia il valore viene elevato al quadrato e quindi da 1). Quindi la sommatoria ci restituirà k_i, il grado del nodo i.
Ma allora la sommatoria $\sum_k B_{ki} B_{kj}$ rappresenta tutti i valori della matrice Laplaciana, ovvero:

$$\sum_k B_{ki} B_{kj} = L_{ij}$$

perché ci da k_i quando ci troviamo sulla diagonale ($i = j$) e -1 se esiste un arco fra i e j (con $i \neq j$), altrimenti ci dà zero. In forma matriciale possiamo scrivere la sommatoria come:

$$L = B^T B$$

Con B^T la trasposta di B. Riprendiamo ora in mano v_i, l'autovettore di L con associato l'autovalore λ_i. Possiamo scrivere sfruttando la definizione di autovalore:

$$v_i^T B^T B v_i = v_i^T L v_i = \lambda_i v_i^T v_i = \lambda_i$$

Per l'ultimo passaggio abbiamo supposto che v_i sia normalizzato e quindi $v_i^T v_i = 1$. L'equivalenza che abbiamo quindi ottenuto è:

$$\lambda_i = v_i^T B^T B v_i$$

cioè ogni autovalore della matrice Laplaciana è equivalente a $(v_i^T B^T)(B v_i)$. Ma questa quantità è di per sé il prodotto vettoriale del vettore $(B v_i)$ con se stesso. Stiamo dicendo che quel valore è la somma delle radici degli elementi (reali) del vettore e quindi non può essere negativo! Abbiamo ottenuto che:

$$\lambda_i \geq 0$$

per ogni i.

Questo risultato è molto importante soprattutto se lo usiamo in merito all'equazione:

$$a_i(t) = a_i(0) \cdot e^{-C\lambda_i t}$$

che avevamo trovato. Infatti stiamo dicendo che e sarà sempre elevato ad un negativo (oppure a 0) e quindi con $t \to \infty$ abbiamo *convergenza* e non divergenza.

Sappiamo ora che ogni autovalore deve essere positivo, ma sappiamo anche (lo andiamo a dimostrare ora) che *uno di essi è zero*.
Consideriamo un vettore unità $\mathbb{1} = (1, 1, 1, \dots)$. Se moltiplichiamo questo vettore con la matrice Laplaciana, otteniamo:

$$\sum_j L_{ij} \cdot 1 = \sum_j (\delta_{ij} k_i - A_{ij}) = k_i - \sum_j A_{ij} = k_i - k_i = 0$$

(abbiamo usato la definizione alternativa degli elementi della Laplaciana ed abbiamo portato fuori il grado che comunque sarebbe stato calcolato nella sommatoria solo quando $j = i$, cioè una volta sola). In notazione matriciale, abbiamo appena scritto che:

$$L\mathbb{1} = 0$$

ma usando la definizione di autovettore è come se scrivessimo:

$$L\mathbb{1} = 0\mathbb{1} = \lambda_1 v_1$$

e quindi abbiamo che il vettore unità è sempre un autovettore della matrice L con autovalore pari a 0. Ordinando gli autovalori da a d n in modo crescente abbiamo quindi che il più piccolo, λ_1, è 0 (non possono essere negativi).
Questa scoperta è notevole poiché ci dice che la matrice Laplaciana non ha inverso, il suo determinante è diverso da zero (il determinante è il prodotto degli autovalori) e quindi L è una matrice singolare.
D'altronde il fatto che λ_1 sia 0 significa che esiste un certo nodo i per cui:

$$a_i(t) = a_i(0) \cdot e^0 \Rightarrow a_i(t) = a_i(0)$$

ovvero per cui in qualsiasi istante la quantità di flusso è sempre la stessa (ed è quella iniziale moltiplicata per l'autovettore di tutti 1, cioè proprio $a(0)$). Ma questo non ci stupisce: quel nodo è il nodo da cui parte l'informazione o che comunque non muta il suo stato (l'individuo zero in una malattia ad esempio!).

9.4 Calcolo delle componenti e algebraic connectivity

Terminiamo i nostri ragionamenti considerando le implicazioni che ha la matrice Laplaciana in merito alla connettività. Consideriamo una rete che è suddivisa in

c componenti di dimensioni $n_1, n_2, ..., n_c$. Ricordiamo che le componenti sono gruppi di nodi sconnessi totalmente da altri gruppi. Per comodità penseremo che i primi n_1 vertici fanno parte della prima componente, i secondi n_2 della seconda, ecc. Ordinando in questo modo i nodi avremo una matrice 'a blocchi', poiché le componenti (totalmente separate fra loro) saranno agglomerate in certe zone, cioè i primi n_1 nodi saranno connessi fra loro, poi i secondi n_2, e così via. La matrice appare in questa forma:

$$\mathbf{L} = \begin{pmatrix} \square & 0 & \cdots \\ 0 & \square & \cdots \\ \vdots & \vdots & \ddots \end{pmatrix}.$$

Figura 9.1: Forma della Laplaciana.

Il primo quadrato, ad esempio, sarà di dimensione $n_1 \times n_1$ ed avrà sempre i vari k_i sulla diagonale e i vari -1 tutto intorno ma ciò che è fondamentale capire è ci sono tutti zero oltre l'n_1-esima colonna per le prime n_1 righe, così come vi sono tutti zero oltre l'n_1-esima riga per le prime n_1 colonne (tutti i valori, che poi rappresentano gli archi, si trovano nel quadrato poiché la componente è limitata ai primi n_1 nodi).
Lo stesso discorso vale per gli altri quadrati, cioè le altre zone (componenti).
Ma allora possiamo immediatamente scrivere c diversi autovettori di L che hanno autovalori zero: sono i vettori che hanno 1 nelle posizioni corrispondenti ai nodi facenti parte della componente e zero altrimenti. Per esempio il vettore:

$$v = (1, 1, 1, ..., 0, 0, ...)$$

che avrà esattamente n_1 uni all'inizio e poi tutti zero. Oppure possiamo considerare questo:

$$v = (0, 0, ..., 1, 1, 1, ..., 0, 0, ...)$$

che avrà n_1 zeri, poi n_2 uni e poi tutti zeri. Tutti questi c diversi vettori sono autovettori di L e sono associati ad un autovalore zero.
Ma stiamo quindi dicendo che se vi sono c componenti, la matrice Laplaciana avrà (almeno) c autovettori che sono associati ad autovalori a zero, cioè c autovalori uguali a zero. Questa riflessione è fondamentale per poter studiare, molto velocemente, in quante componenti è suddiviso un grafo. In realtà si può dimostrare che il numero di autovalori a zero ci dà esattamente il numero di componenti (infatti, se la componente è unica abbiamo solamente il primo autovalore che abbiamo dimostrato essere sempre a zero). Per questa ragione il secondo autovalore più piccolo (λ_2 secondo l'ordinamento che avevamo imposto prima) è molto importante: se è diverso da zero ci dice che

il grafo è connesso. Per questa ragione tale valore è detto *algebraic connectivity* della rete. Questo valore salta fuori negli algoritmi che si occupano di *partizionamento spettrale* (di cui abbiamo parlato nel corso di questo volume).

Bibliografia

[1] Mark E. J. Newman. *Networks: An Introduction*. Oxford University Press, 2010.