

Teoria dei grafi e algoritmi su grafi

Algoritmi per l'intelligenza artificiale

Vincenzo Bonnici

Corso di Laurea Magistrale in Scienze Informatiche

Dipartimento di Scienze Matematiche, Fisiche e Informatiche

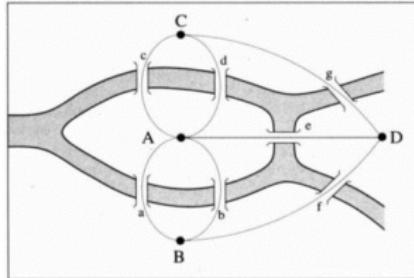
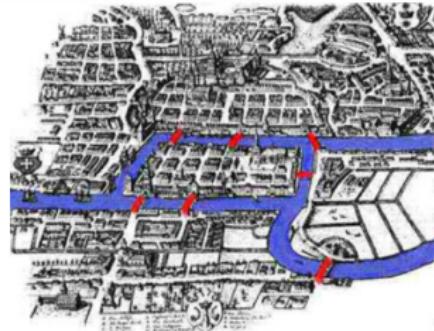
Università degli Studi di Parma

2025-2026

I sette ponti di Königsberg

La **teoria dei grafi** ha una data di nascita precisa: il 1736. In quella data, il matematico svizzero Leonhard **Euler** risolse il problema noto come i **sette ponti di Königsberg**.

Ci si chiedeva se fosse possibile fare una passeggiata in città, che partisse e arrivasse allo stesso punto, in modo da attraversare tutti i ponti esattamente una volta.



William Hamilton nacque a Dublino nel 1805. Fu astronomo, fisico e matematico: a lui si deve la definizione dei quaternioni.

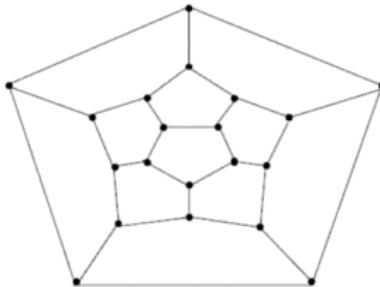
Nel 1859 Hamilton propose un gioco che, per diversi aspetti, era legato alla sua teoria dei quaternioni: lo cedette per 25 sterline a un commerciante (quello fu l'unico denaro che ricevette direttamente per una scoperta o pubblicazione, ci dice il suo biografo).

Il gioco dell'icosaedro (che in realtà si giocava su un dodecaedro) era il seguente: Hamilton aveva assegnato a ogni vertice il nome di una città e richiedeva di trovare un percorso che facesse il giro del mondo, ossia visitasse tutte le città una sola volta, per poi tornare al vertice di partenza.

Il gioco dell'icosaedro

Se immaginiamo che la superficie di un dodecaedro sia fatta di gomma, possiamo bucare una delle sue facce e stirarla su una superficie: allora gli spigoli formeranno la rete mostrata in figura.

Su un dodecaedro con vertici non segnati sono possibili solo due circuiti di Hamilton, immagini speculari l'uno dell'altro. Se però i vertici vengono marchiati e consideriamo ogni percorso differente se passa per i 20 vertici in ordine diverso, allora i circuiti diversi diventano 30 (non contando le sequenze uguali percorse in senso inverso).



Il problema del commesso viaggiatore

Una variante del gioco dell'icosaedro è il **problema del commesso viaggiatore (TSP)**.

Si tratta di trovare il percorso chiuso più breve in un grafo completo pesato, ossia i cui lati hanno lunghezze diverse.

Questo è il problema per eccellenza nella ottimizzazione combinatoria. Non è un problema trovare un circuito chiuso: in un grafo completo a n nodi esistono $1/2(n - 1)!$ circuiti chiusi. Il problema è trovare il migliore. Questo problema è contenuto nella classe dei problemi **NP-hard**.

Il primo passo importante fu compiuto da George Dantzig nel 1954 che pubblicò la descrizione di un algoritmo per risolvere il TSP e lo applicò al problema di trovare il cammino minimo tra 49 città.

Sei gradi di Kevin Bacon

In una intervista del gennaio 1994, l'attore Kevin Bacon affermò di avere lavorato con tutti gli attori di Hollywood o, almeno, con qualcuno che ci aveva recitato insieme.

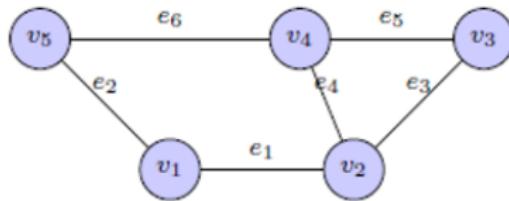
Ispirato da questa notizia, il 7 aprile 1994 uscì un lunghissimo articolo su un newsgroup intitolato Kevin Bacon is the Center of the Universe. Poco dopo, tre studenti di college crearono il gioco **Six Degrees of Kevin Bacon**, seguito a breve da un libro e da un sito web, The Oracle of Bacon, che calcola il numero di Bacon di ogni attore presente nel database IMD (Internet Movie database).



- Il calcolo del numero di Bacon per un attore si basa sulla ricerca di un cammino minimo:
 - Kevin Bacon ha numero di Bacon 0;
 - gli attori che hanno collaborato direttamente con lui hanno numero di Bacon 1;
 - se tra i collaboratori di un attore X il minimo numero di Bacon è n , allora il numero di Bacon di X è $n + 1$
- nel febbraio 2013, il numero di Bacon finito più alto era 11;
- allo stesso tempo, circa il 12% delle persone citate nel database non erano connessi con Kevin Bacon in nessun modo;

Kevin Bacon non è il centro dell'universo di Hollywood, se per centro dell'universo intendiamo l'attore con la distanza media minima da tutte le altre persone a lui connesse. Il numero di persone connesse con ogni attore e le loro distanze variano con l'uscita di ogni film e il centro dell'universo cambia di conseguenza. Secondo il sito web The Oracle of Bacon, ad Aprile 2013 il centro dell'universo tra Harvey Keitel.

Un **grafo** $G = (V, E)$ è un insieme di **vertici (o nodi)** V e un insieme di **lati (o archi)** E , che per semplicità supporremo finiti, quindi $|V| = n$ e $|E| = m$.



In questo esempio, $V(G) = \{v_1 \dots v_5\}$ e $E(G) = \{e_1 \dots e_6\}$. Il lato $e_k = (v_i, v_j)$ è detto **incidente** ai vertici v_i e v_j .

Il **grado** di un vertice $v \in V$ è il numero $d(v)$ dei lati incidenti.

Un self-loop conta 2 nel calcolo del grado di un vertice.

Un vertice isolato ha grado 0.

Teorema (lemma delle strette di mano)

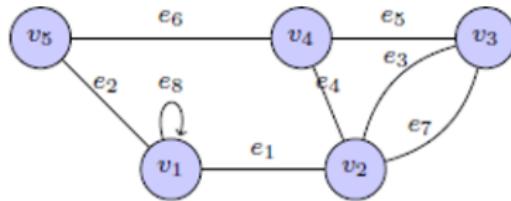
La somma dei gradi di un grafo $G = (V, E)$ vale $d(V) = 2|E| = 2m$.
(banale)

Corollario: Il numero di vertici di grado dispari è pari

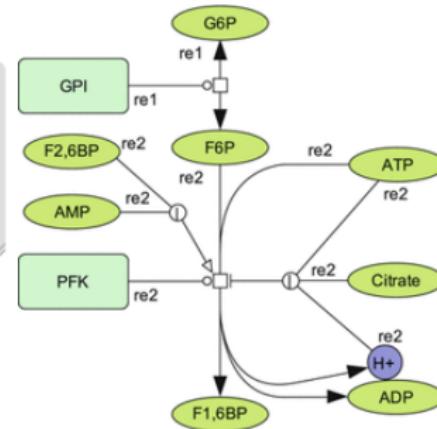
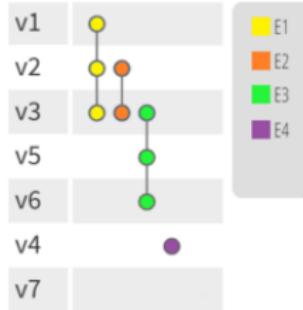
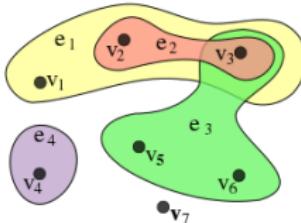
Questo corollario è certamente degno di nota, perché rappresenta un vincolo a cui deve sottostare ogni rete: ad esempio, non è possibile mettere insieme nove persone in modo tale che ognuna di loro ne conosca esattamente altre cinque del gruppo.

Teoria dei grafi

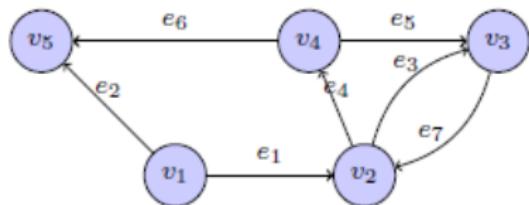
Il grafo precedente è detto semplice perché non contiene self-loop né lati multipli. Questo tipo di grafi viene anche detto **multigrafo**.



Esistono poi grafi in cui un singolo arco può incidere su più di due vertici. Tali grafi vengono chiamato **ipergrafo**.



In un **grafo diretto**, ogni lato ha una direzione.



Diremo che nel lato $e = (v_s, v_t)$, v_s è la **partenza (o sorgente)** e v_t è **l'arrivo (o destinazione)**.

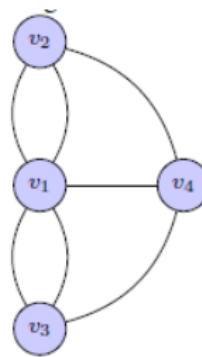
Ogni nodo ha un **grado entrante** $d_{in}(v)$ e un **grado uscente** $d_{out}(v)$.

Un grafo è **bilanciato** se $d_{in}(v) = d_{out}(v)$ per ogni nodo $v \in V$.

Teoria dei grafi

La **distanza** tra due vertici u e v è la lunghezza del **minimo cammino** che si deve attraversare per andare da u a v . Ad esempio, nel grafo di Königsberg, il vertice v_1 ha distanza 1 da ogni altro vertice, ma la distanza tra v_2 e v_3 è pari a 2.

In generale, diremo che due vertici sono adiacenti se hanno distanza 1 (se sono connessi da un lato).

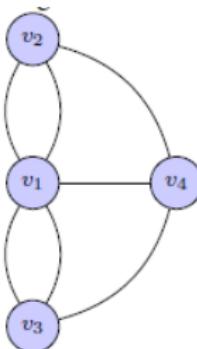


Il **diametro** di un grafo è il massimo delle distanze tra due nodi del grafo. Così, il grafo di Königsberg ha diametro 2; un grafo completo ha sempre diametro 1.

Circuiti euleriani

Un **circuito (o anello)** in un grafo è un cammino che inizia e finisce dallo stesso vertice.

Un **circuito euleriano** è un circuito che passa per ogni lato (arco) del grafo esattamente una volta.



Teorema di Eulero

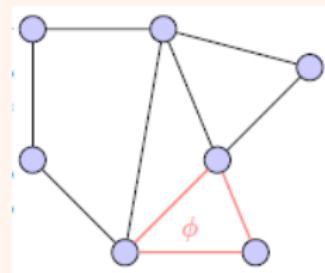
Un grafo ha un circuito euleriano se e solo se ogni vertice ha grado pari.

Necessità

Se esiste un circuito Euleriano, allora questo visita tutti i nodi. Ogni volta che raggiungiamo un nodo v , lo dobbiamo lasciare con un altro lato, quindi il suo grado è pari.

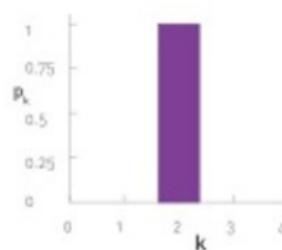
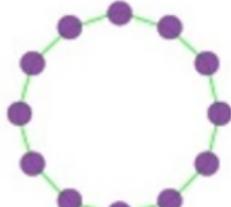
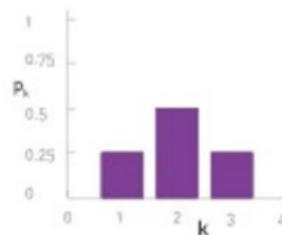
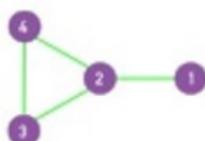
Sufficienza

In G deve esistere un ciclo ϕ . Consideriamo il sottografo H ottenuto rimuovendo da G i lati in ϕ . I nodi in H che non sono isolati hanno ancora grado pari. Allora da ogni componente连通的 di H posso ottenere un ciclo ϕ_i . Continuo finché non riduco tutto il grafo a nodi isolati. Per costruire il circuito euliano, partiamo da ϕ ; ogni volta che incontriamo un nodo che sta in un altro ciclo ϕ_i , aggiungiamo tutto il ciclo ϕ_i al nostro cammino. Alla fine, si ottiene un circuito euliano.



Distribuzione dei gradi

$P(k) = N_k/N$, con N_k numero di nodi con grado k e N numero di nodi totali : probabilità che scegliendo a caso un nodo dal grafo questi abbia grado k , equivalente alla frequenza del grado k all'interno del grafo.



Negli anni '60 dello scorso secolo, lo psicologo sperimentale Stanley Milgram concepì alcuni fondamentali studi sul fenomeno detto **small world** nelle reti sociali dell'uomo. Milgram cercò di stimare la distanza tipica tra due nodi qualunque di una rete sociale, come ad esempio quella degli attori; il suo scopo era dimostrare che, in generale, queste distanze dovevano essere piccole.

I suoi esperimenti avevano lo scopo di mostrare che la globalizzazione rendeva il mondo più piccolo.

6 gradi di separazione

Nel suo esperimento più famoso, Milgram spedì 96 pacchi a altrettante persone che vivevano a Omaha, NE, USA, che aveva scelto a caso dall'elenco telefonico.

Ogni pacchetto conteneva un messaggio, scritto sulla carta intestata dell'Università di Harvard (di cui Milgram era membro), che invitava il destinatario a partecipare all'esperimento e le istruzioni per lo stesso.

Si chiedeva alle persone di cercare di fare avere il pacchetto a una persona (un amico di Milgram, che viveva a Boston, MA, USA). Di questa persona veniva dato solo il nome (quindi il sesso), l'indirizzo e l'occupazione (agente di borsa). Alle persone di Omaha si chiedeva di mandare il messaggio a qualcuno che conoscevano personalmente e che ritenevano (per similitudine di lavoro, oppure per vicinanza geografica, o altro) potesse essere più vicino al bersaglio. Ogni persona che riceveva il pacchetto avrebbe dovuto inoltrarlo, con le stesse regole, semplicemente aggiungendo i dettagli del passaggio.

6 gradi di separazione

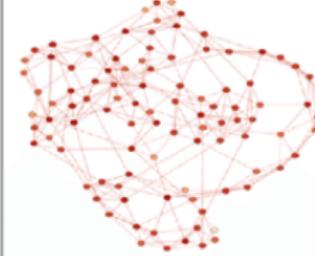
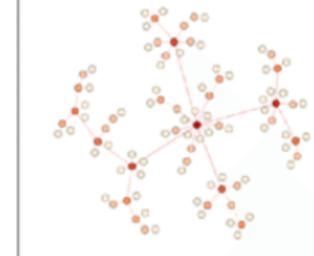
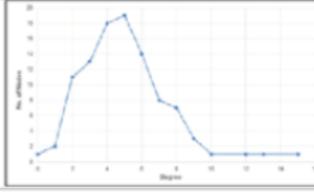
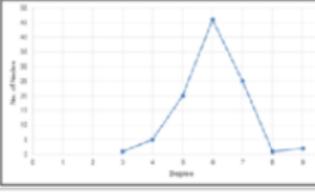
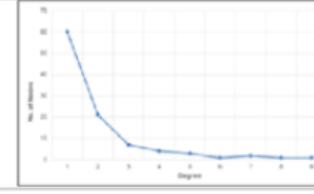
L'amico di Milgram ricevette 18 dei 96 pacchetti. Questo tasso di successo è molto alto, e quando si è tentato di ripetere l'esperimento usando la posta elettronica (Dodds, 2003) non si è assolutamente ripetuto.

Il numero medio di passaggi di mano che aveva avuto ogni pacchetto era di circa 5,9. Questo numero venne poi popolarizzato nella locuzione sei gradi di separazione.

Ancora più particolare, poi, notare come la velocità di trasmissione dell'informazione era creata da una conoscenza in massima parte locale della rete sociale, era analizzata con informazioni soprattutto locali.

Abbiamo già visto come questa idea funzioni nel caso del cinema; sappiamo che vale anche in matematica. Il sito MathSciNet della American Mathematical Society, oltre a contenere un importante catalogo di pubblicazioni di matematica e materie affini, fornisce per ogni autore un **numero di Erdős** che misura la distanza di un matematico dal matematico Paul Erdős, che fu uno dei fondatori della teoria dei grafi.

Tipi di reti

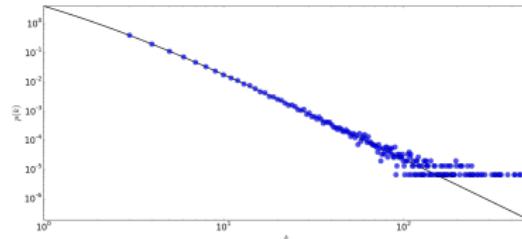
	Random	Small-World	Scale-Free
Visualisation of the Topology			
Degree Distribution			
Robustness Characteristics	Responds similarly to both random and targeted attacks.	Resilient against random failures but very sensitive to targeted attacks.	

Rete a invarianza di scala (scale-free networks)

Viene definita **rete a invarianza di scala** un grafo che gode della seguente proprietà:

Se si considera la relazione tra il numero di nodi ed il numero delle loro connessioni si vede che il suo grafico è di tipo esponenziale negativo, e quindi invariante per cambiamenti di scala.

Questa invarianza di scala significa che paragonando il numero di due tipi di nodi, ad esempio quelli con 10 connessioni e quelli con 15, si vede che la proporzione (N_a/N_b) fra i due è $e^{-a(N_b-N_a)}$, dove N_b ed N_a sono il numero di nodi del denominatore e numeratore mentre a è un parametro del tipo di rete considerato. Questa legge è detta **legge di potenza**, di cui a è il parametro.



Il modello Barabási – Albert (BA) è un algoritmo per la generazione di reti a invarianza di scala casuali utilizzando un meccanismo di attacco preferenziale.

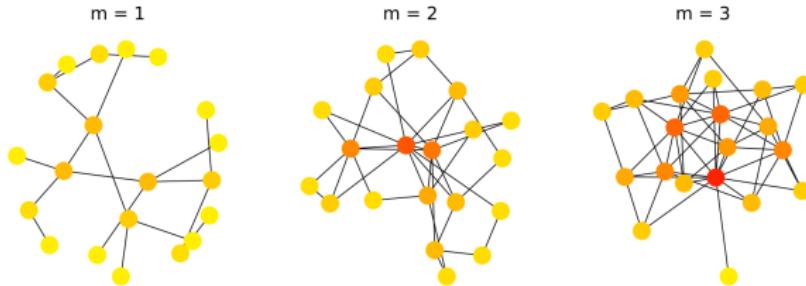
La rete inizia con una rete connessa iniziale di m_0 nodi.

I nuovi nodi vengono aggiunti alla rete uno alla volta. Ogni nuovo nodo è connesso a $m \leq m_0$ nodi esistenti con una probabilità proporzionale al numero di collegamenti che i nodi esistenti hanno già.

Formalmente, la probabilità p_i che il nuovo nodo sia connesso al nodo i è:

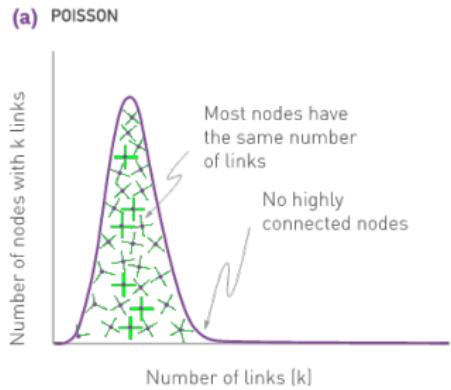
$$p_i = \frac{k_i}{\sum_j k_j}$$

dove k_i è il grado del nodo i e la somma è fatta su tutti i nodi preesistenti j (cioè il denominatore risulta doppio dell'attuale numero di archi nella rete).

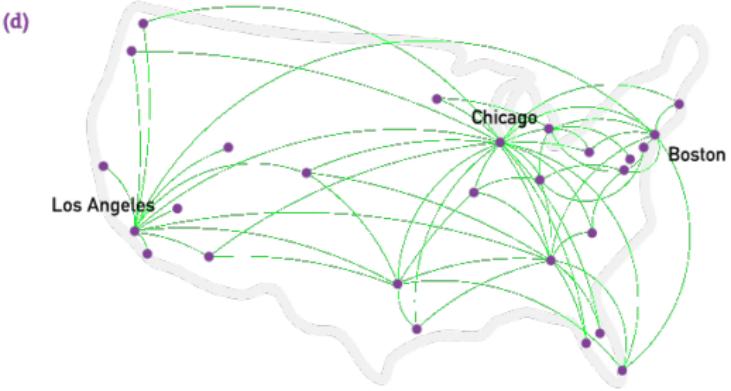
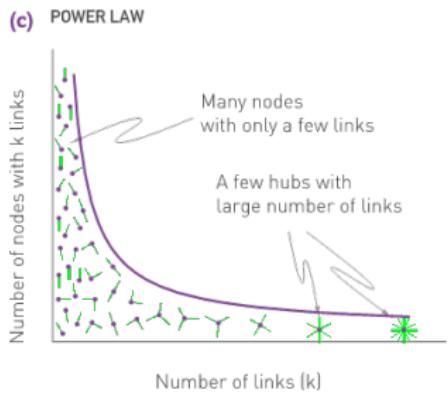


I nodi fortemente collegati ("hub") tendono ad accumulare rapidamente ancora più collegamenti, mentre è improbabile che i nodi con pochi collegamenti vengano scelti come destinazione per un nuovo collegamento. I nuovi nodi hanno una "preferenza" per attaccarsi ai nodi già fortemente collegati.

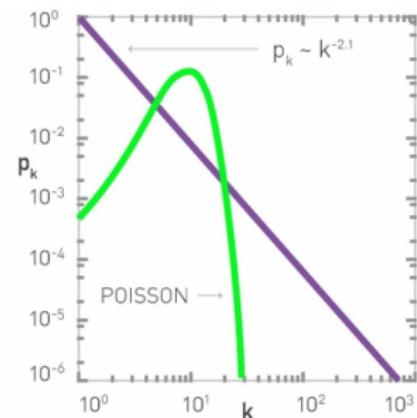
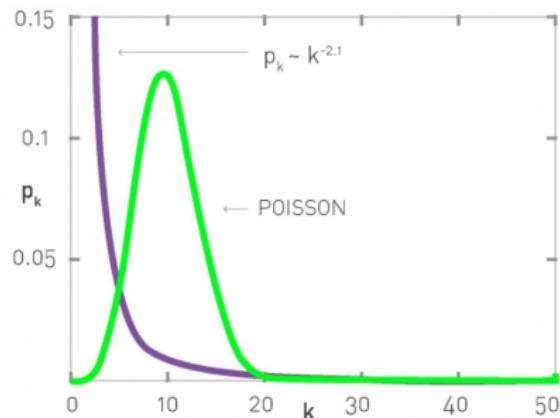
Differenze tra power-law e Poisson



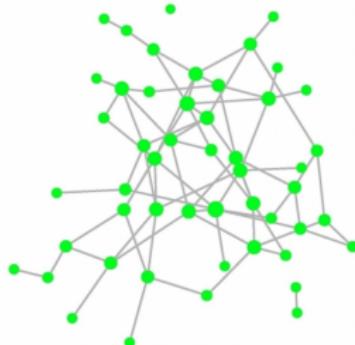
Differenze tra power-law e Poisson



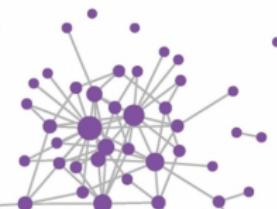
Differenze tra power-law e Poisson



c.



d.



Nel modello di Erdős e Rényi, tutti i grafi su un insieme di vertici fisso con un numero fisso di archi sono ugualmente probabili.

Nel modello introdotto da Gilbert, chiamato anche modello Erdős–Rényi–Gilbert, ogni arco ha una probabilità fissa di essere presente o assente, indipendentemente dagli altri archi.

Questi modelli possono essere utilizzati nel metodo probabilistico per dimostrare l'esistenza di grafi che soddisfano varie proprietà o per fornire una definizione rigorosa di cosa significhi che una proprietà vale per quasi tutti i grafi.

Nel modello $G(n, M)$, un grafo viene scelto uniformemente a caso dalla raccolta di tutti i grafici che hanno n nodi e M bordi. I nodi sono considerati etichettati, il che significa che i grafici ottenuti l'uno dall'altro permutando i vertici sono considerati distinti.

Ad esempio, nel modello $G(3, 2)$, ci sono tre grafi a due spigoli su tre vertici etichettati (uno per ogni scelta del vertice medio in un percorso a due archi), e ciascuno di questi tre grafi è incluso con probabilità $\frac{1}{3}$.

Nel modello $G(n, p)$, viene costruito un grafo collegando casualmente nodi etichettati. Ogni arco è incluso nel grafo con probabilità p , indipendentemente da ogni altro arco. Equivalentemente, la probabilità di generare ogni grafo che ha n nodi e M archi è $p^M(1 - p)^{\binom{n}{2} - M}$.

Il parametro p in questo modello può essere pensato come una funzione di ponderazione; man mano che p aumenta da 0 a 1, il modello diventa sempre più probabile che includa grafi con più bordi ed ha sempre meno probabilità di includere grafi con meno bordi. In particolare, il caso $p = \frac{1}{2}$ corrisponde al caso in cui tutti i $2^{\binom{n}{2}}$ grafi su n vertici sono scelti con uguale probabilità.

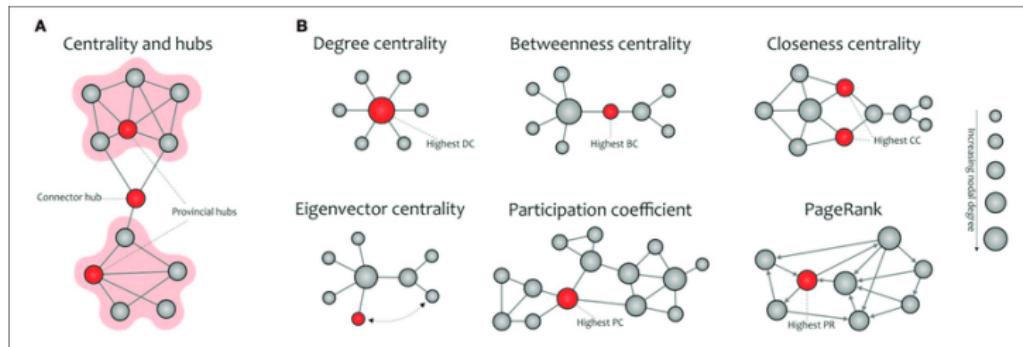
Nella **teoria della percolazione** si esamina un grafo finito o infinito e si rimuovono i bordi (o collegamenti) in modo casuale. Quindi il processo Erdős-Rényi è di fatto una percolazione del collegamento non pesata sul grafo completo.

La centralità è un concetto che tipicamente viene applicati ai **vertici** (o archi) di un grafo per descriverne una caratteristica di **importanza topologica** di tale vertice rispetto al resto del grafo, a livello si **locale** che **globale**.

La centralità può essere utilizzata non solo per studiare staticamente le proprietà di un grafo ma per descrivere come un grafo potrebbe **cambiare** nel tempo e quali **conseguenze** avrebbe l'inserimento o la cancellazione di vertici o archi dal grafo.

Una applicazione diffusa è la ricerca dei vertici **hub** quando questi si dimostrano essere elementi essenziali per il funzionamento del **sistema**.

Centralità



Centrality name	Description	Refs.
Betweenness Centrality (BC)	$C_B(u) = \sum_{s \neq u \neq t} \frac{\rho(s,u,t)}{\rho(s,t)}$ is the total number of shortest paths from node s to node t $\rho(s,u,t)$ is the number of those paths that pass through u	Anthonisse, 1971
Closeness Centrality (CC)	$C_C(u) = \frac{ N_u - 1}{\sum_{v \in N_u} \text{dist}(u,v)}$ is the number of node u 's neighbors. $\text{dist}(u,v)$ is the distance of the shortest path from node u to node v .	Sabidussi, 1966
Degree Centrality (DC)	$C_D(u) = N_u $ $ N_u $ is the number of node u 's neighbors.	Jeong et al., 2001
Eigenvector Centrality (EC)	$C_E(u) = \alpha_{\max}(A)\mu_{\max}$ is the eigenvector corresponding to the largest eigenvalue. of the adjacency matrix A 's.	Bonacich, 1987
Local average connectivity-based method (LAC)	$LAC(u) = \frac{\sum_{v \in N_u} \deg_{cu}(v)}{ N_u }$ C_u is the subgraph induced by N_u , $\deg_{cu}(v)$ is the number of nodes is directly connected in C_u	Li et al., 2011
Network Centrality (NC)	$C_N(u) = \sum_{v \in N_u} ECC(u, v) \sum_{v \in N_u} \frac{ N_u \cap N_v }{\min(N_u , N_v - 1)} N_u$ is the node set containing all the neighbors of node u . is the node set containing all the common neighbors of node u and node v .	Wang et al., 2012
Subgraph Centrality (SC)	$C_S(u) = \sum_{k=0}^{\infty} \frac{\mu_k(u)}{k!} = \sum_{v=1}^N (\nu_k^u)^2 e^{\lambda_k} \mu_k(u)$ is the u th diagonal entry of the k th power of the adjacency matrix of network. $\nu_1, \nu_2, \dots, \nu_N$ is R^N be an orthonormal basis of composed by eigenvectors of A associated to the eigenvalues $\lambda_1, \lambda_2, \dots, \lambda_N$ ν^u is the u th component of ν_u .	Estrada and Rodriguez-Velazquez, 2005
Information Centrality (IC)	$I_{u,v} = (c_{u,u} + c_{v,v} - 2 \times c_{u,v})^{-1} I_{u,v} = (c_{u,u} + c_{v,v} - 2 \times c_{u,v})^{-1} C = (c_{u,v}) = (D - A + J)^{-1} C = A$ is the adjacency matrix of network, D is the diagonal matrix of all nodes' degree, J is a matrix whose all elements are 1.	Stephenson and Zelen, 1989

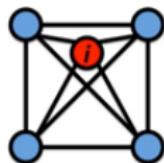
Centralità

Centrality measures defined on weighted graph (Li et al., 2013).

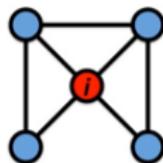
Centrality name	Description
Betweenness Centrality (with weight)	$\text{cost}(s,t) = \frac{1}{w_{s,t}} \times w_{s,t} > 0$ $w_{s,t}$ is the weight of edge from node s to node t , $\text{dist}_w(u,v) = \sum_{(s,t) \in P} \text{cost}(s,t)$ The shortest path from node s to node t is the path whose $\text{dist}_w(u,v)$ is smallest. $C_{B_w}(u) = \sum_{s \neq u \neq t} \frac{\rho(s,u,t)}{\rho(s,t)} \rho(s,t)$ is the total number of shortest paths from node s to node t . $\rho(s,u,t)$ is the number of those paths that pass through u .
Closeness Centrality (with weight)	$\text{cost}(s,t) = \frac{1}{w_{s,t}} > 0$ $w_{s,t}$ is the weight of edge from node s to node t , $\text{dist}_w(u,v) = \sum_{(s,t) \in P} \text{cost}(s,t) \text{dist}_w(u,v)$ is the weighted distance of the shortest path from node u to node v . $C_{C_w}(u) = \frac{ N_u - 1}{\sum_{v \in N_u} \text{dist}_w(u,v)}$ is the number of node's neighbors.
Degree Centrality (with weight)	$C_{D,w}(u) = \sum_{v \in N_u} w(u,v) N_u$ is the node set containing all the neighbors of node u . $w(u,v)$ is the weight of the edge connected node u and node v .
Eigenvector Centrality (with weight)	$C_E(u) = \alpha_{\max}(u) \alpha_{\max}$ is the eigenvector corresponding to the largest eigenvalue of the weighted adjacency matrix A_w .
Local average connectivity-based method (with weight)	$LAC_w(u) = \sum_{s \in N_u} \sum_{t \in N_u} \frac{w(s,t)}{ N_u } N_u$ is the node set containing all the neighbors of node u . $w(s,t)$ is the weight of the edge connected node S and node t .
Network Centrality (with weight)	$C_{N_w}(u) = \sum_{v \in N_u} \frac{\sqrt{\sum_{s \in N_u \cap N_v} w(u,s) \times \sum_{t \in N_u \cap N_v} w(v,t)}}{\min(\sum_{s \in N_u} w(u,s), 1, \sum_{t \in N_v} w(v,t), 1)} N_u$ is the node set containing all the neighbors of node u . $ N_u \cap N_v $ is the node set containing all the common neighbors of node u and node v . $w(u,s)$ is the weight of the edge connected node u and node s .
Subgraph Centrality (with weight)	$C_{S_w}(u) = \sum_{l=0}^N \frac{\mu_l(u)}{\mu_l^N} = \sum_{v=1}^N (v_w^u)^2 e^{k_w} \mu_l(u)$ is the u th diagonal entry of the l th power of the weighted adjacency matrix of network. v_1, v_2, \dots, v_N is be an orthonormal basis of R^N composed by eigenvectors of A associated to the eigenvalues $\lambda_1, \lambda_2, \dots, \lambda_N$. v_w^u is the u th component of v_w .
Information Centrality (with weight)	$C_{I_w}(u) = \left[\frac{1}{ V } \sum_{v \in V} \frac{1}{I_{u,v}} \right]^{-1} I_{u,v} = (c_{u,u} + c_{v,v} - 2 \times c_{u,v})^{-1} (c_{u,v}) = (D_w - A_w + J_w)^{-1} I_{u,v} = 0$ A is the weighted adjacency matrix of network, D is the diagonal matrix of all nodes' degree, J is a matrix where all elements are 1.

Coefficiente di clustering

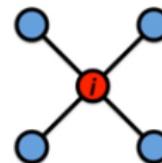
Quale è la frazione di vicini di un nodo che sono connessi tra di loro?



$$C_i = 1$$



$$C_i = 1/2$$



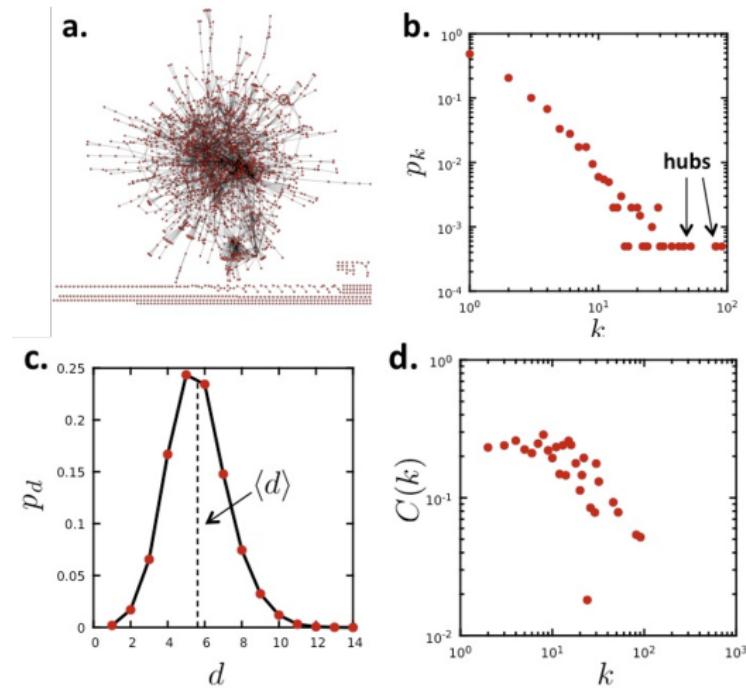
$$C_i = 0$$

Dato un grafo $G = (V, E)$, dato un vertice $v_i \in V$ il suo coefficiente di clustering C_i è dato dalla porzione del numero di archi tra i vertici nel suo vicinato diviso il numero possibile totale di archi tra tali vertici:

$$C_i = \frac{|\{e_{jk} : v_j, v_k \in N_i, e_{jk} \in E\}|}{d_i(d_i - 1)}$$

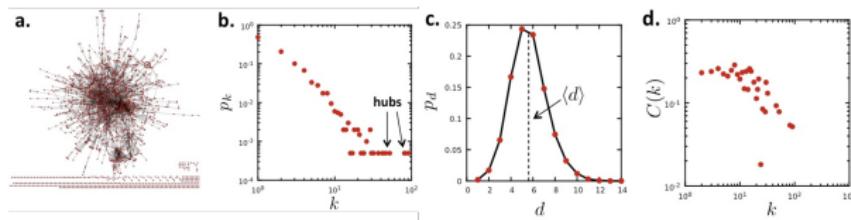
con $N_i = \{v_j : e_{ij} \in E\}$ l'insieme dei vicini di v_i .

Coefficiente di clustering



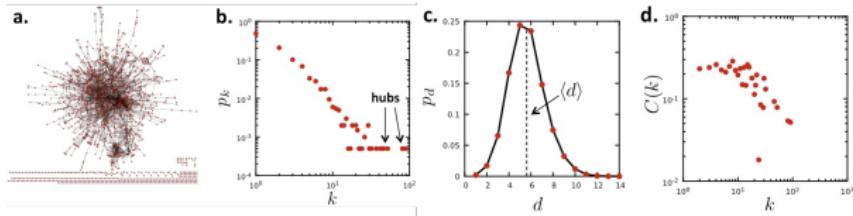
(a) Rete di interazione fisica proteina-proteina. (b) distribuzione dei gradi.
(c) distibuzione delle lunghezze dei cammini minimi. (d) Coefficiente di clustering medio per nodi di grado k .

Coefficiente di clustering



Un'ispezione visiva rivela uno schema interessante: gli hub hanno la tendenza a connettersi a piccoli nodi, conferendo alla rete un carattere hub and spoke. Questa è una conseguenza delle correlazioni di grado, che riflettono la natura disordinativa della rete di interazione proteica. Le correlazioni di grado influenzano una serie di caratteristiche della rete, dalla diffusione di idee e virus nelle reti sociali al numero di nodi driver necessari per controllare una rete.

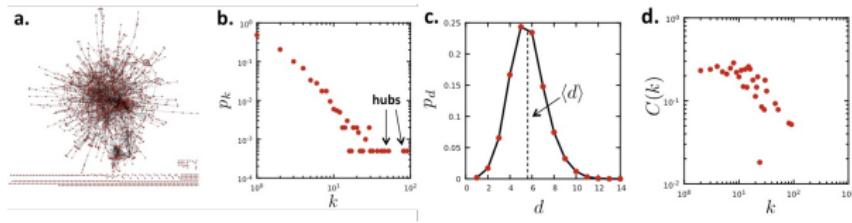
Coefficiente di clustering



La distribuzione dei gradi indica che la stragrande maggioranza dei nodi ha solo pochi collegamenti. Per la precisione, in questa rete il 69% dei nodi ha meno di tre collegamenti, cioè per questi k minori del grado medio. Coesistono con pochi nodi altamente connessi, o hub, il più grande dei quali ha ben 91 collegamenti.

Tali ampie differenze nei gradi dei nodi sono una conseguenza della proprietà di assenza di scala della rete, una proprietà che si incontra in molte reti reali.

La forma della distribuzione dei gradi determina un'ampia gamma di proprietà della rete, dalla robustezza di una rete ai guasti dei nodi alla diffusione di virus.

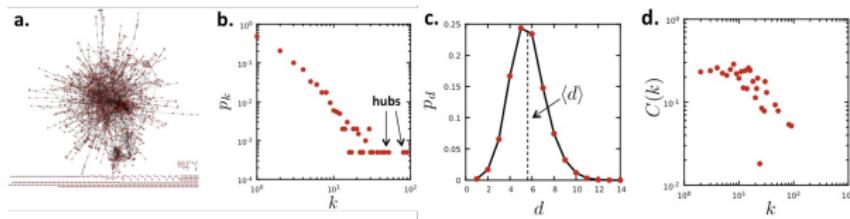


Se utilizziamo l'algoritmo di ricerca in ampiezza, possiamo determinare il diametro della rete, trovando $d_{max} = 14$.

Potremmo essere tentati di aspettarci ampie variazioni in d , poiché alcuni nodi sono vicini l'uno all'altro, altri, tuttavia, forse abbastanza lontani. La distribuzione delle distanze indica diversamente: p_d ha un picco prominente intorno a $\langle d \rangle = 5.61$, indicando che la maggior parte delle distanze sono piuttosto brevi, essendo nelle vicinanze di $\langle d \rangle$. Inoltre, p_d decade velocemente per grandi $\langle d \rangle$, suggerendo che le grandi distanze sono essenzialmente assenti.

Queste sono manifestazioni della proprietà di small world, un'altra caratteristica comune delle reti reali, che indica che la maggior parte dei nodi sono vicini l'uno all'altro.

Coefficiente di clustering

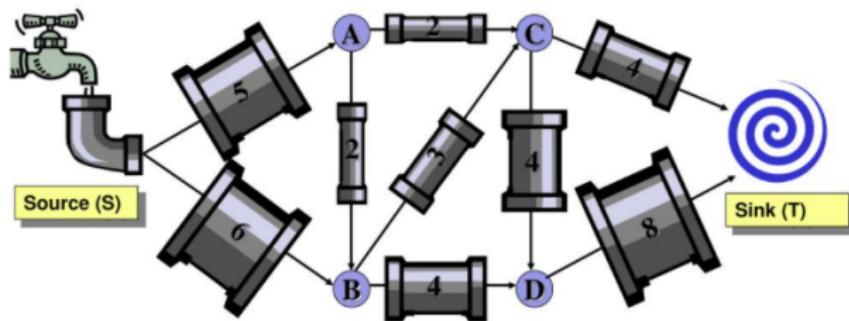


Il coefficiente medio di clustering della rete è $\langle C \rangle = 0,12$, che, come capiremo, è piuttosto grande, indicando un grado significativo di clustering locale nella rete.

Un ulteriore evidenza è fornita dalla dipendenza del coefficiente di clustering dal grado del nodo, o $C(k)$, che indica che il coefficiente di clustering dei nodi piccoli è significativamente più alto del coefficiente di clustering degli hub. Ciò suggerisce che i nodi di grado piccolo tendono a far parte di quartieri locali densi, mentre il vicinato dei nodi hub è molto più sparso.

Questa è una conseguenza della gerarchia di rete, un'altra proprietà di rete ampiamente condivisa.

Un tipo di problema che viene spesso analizzato utilizzando le tecniche della teoria dei grafi è il problema dei **flussi sui grafi**. La rete può rappresentare un sistema stradale e il flusso rappresenta il movimento dei veicoli da un punto all'altro sulle strade. Oppure il network può essere una rete di computer, dove chi si muove sono i pacchetti dati da un computer all'altro. Oppure una rete idrica, postale o altro.

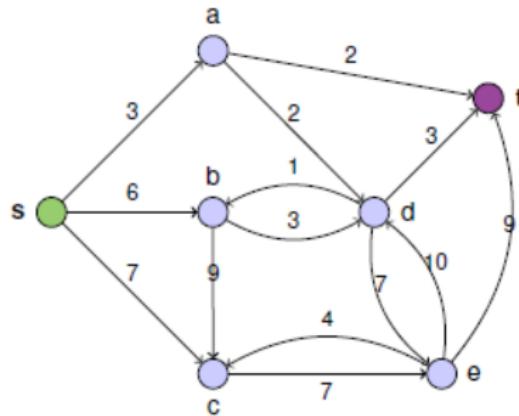


Il network che consideriamo è dato da un grafo diretto $G = (V, E)$ con un nodo **sorgente** s (con grado uscente $d_{out}(s) > 0$) e un nodo **terminale** t con grado entrante $d_{in}(t) > 0$.

Inoltre, ad ogni lato associamo una **capacità** positiva $c(e) > 0$. La capacità di un lato determina quanto materiale (veicoli, pacchetti, energia, acqua) può passare lungo il lato in una data unità di tempo. Così, strade piccole o piccole condotte avranno bassa capacità, mentre condotte o strade larghe hanno alta capacità.

Per semplicità, supporremo che le capacità siano **numeri interi**. In queste note, saremo interessati in un problema piuttosto semplice, quello di **massimizzare il trasporto di materiale dalla sorgente al terminale**. Per semplicità, supporremo che non esistano lati entranti nella sorgente e lati uscenti dal terminale.

Questo è un esempio di sistema di distribuzione dell'acqua, dove i lati sono le condotte, i vertici sono stazioni di pompaggio e la direzione di un lato indica in che direzione le nostre pompe spingono l'acqua. La capacità di ogni lato rappresenta il massimo numero di metri cubi al secondo di acqua che possiamo immettere in ogni condotta.



Il nostro scopo sarà quello di muovere la maggiore quantità possibile di acqua dalla sorgente al terminale, sottostando alle seguenti condizioni:

- possiamo muovere l'acqua sempre nella **direzione** della condotta
- non possiamo immettere in una condotta più acqua di quanta sia la sua **capacità**
- in una stazione intermedia tra sorgente e terminale, il **flusso totale** entrante deve essere uguale al flusso totale uscente

Un **flusso ammissibile** f è una funzione sui lati $e = (v_i, v_j) \in E$ tale che

- per ogni lato e si ha $0 \leq f(e) \leq c(e)$

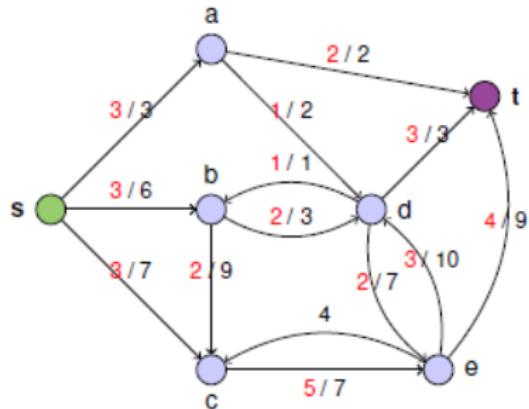
- per ogni vertice $v \in V \setminus \{s, t\}$ vale

$$\sum_{u \in V | (u,v) \in E} f(u, v) = \sum_{z \in V | (v,z) \in E} f(v, z).$$

Il **flusso totale** del network è allora pari al **flusso uscente** dalla sorgente e al **flusso entrante** al terminale:

$$F(G) = \sum_{u \in V | (s,u) \in E} f(s, u) = \sum_{z \in V | (z,t) \in E} f(z, t)$$

Diamo un esempio di flusso sulla rete vista precedentemente, dove indichiamo in rosso il flusso assegnato ad ogni nodo.



Questo flusso è ammissibile, in quanto ogni nodo è in pareggio e nessuna condotta è utilizzata oltre la sua capacità. Il flusso totale del network è pari a $3 + 3 + 3 = 4 + 3 + 2 = 9$. Ci domandiamo allora se è possibile fare di meglio, ossia aumentare il flusso totale del network.

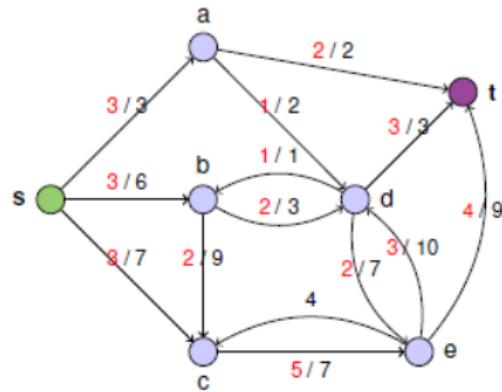
Parleremo di **cammini migliorabili** se, in un network con un flusso ammissibile, considerando il network come grafo non diretto, esiste un cammino $\{e_1 \dots e_n\}$ che congiunge s a t tale che

- ogni lato e_k che punta in avanti (verso t) ha un flusso associato inferiore alla capacità massima, e
- ogni lato e_j che punta indietro (verso s) ha un flusso associato positivo.

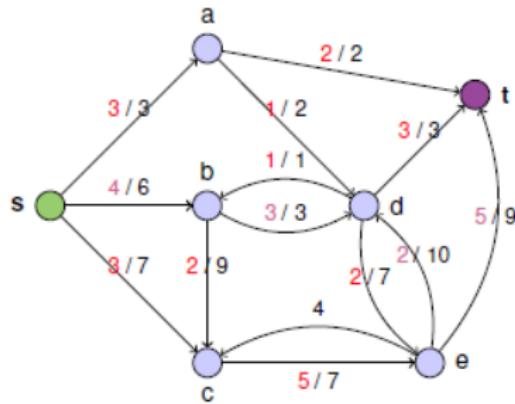
Se abbiamo a disposizione un cammino migliorabile possiamo aumentare il flusso totale del network:

- si aumenta di una unità il flusso su tutti i lati che puntano in avanti
- si diminuisce di una unità il flusso su tutti i lati che puntano indietro.

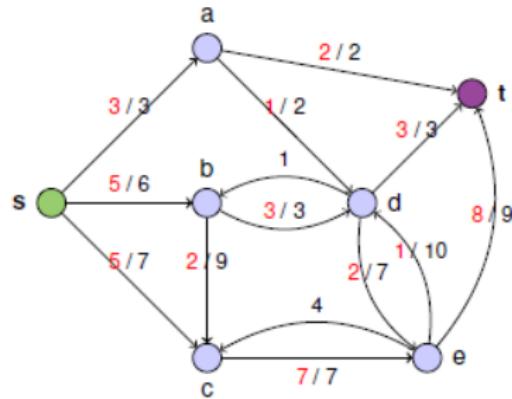
Diamo un esempio di cammino migliorabile sulla rete vista precedentemente.



Si prenda il cammino $(\vec{s}, \vec{b}), (\vec{b}, \vec{d}), (\vec{d}, \vec{e}), (\vec{d}, \vec{e}), (\vec{e}, \vec{t})$. È un cammino migliorabile perché tutti i flussi in avanti sono aumentabili (ad esempio, il primo vale $3 / 6$ e tutti i flussi indietro sono positivi (il flusso su (\vec{d}, \vec{e}) vale $3 / 10$). Usando l'algoritmo descritto in precedenza possiamo allora costruire un flusso ammissibile.



Ripetendo l'algoritmo, otteniamo ad un certo punto questo flusso, che non ammette altri cammini migliorabili.



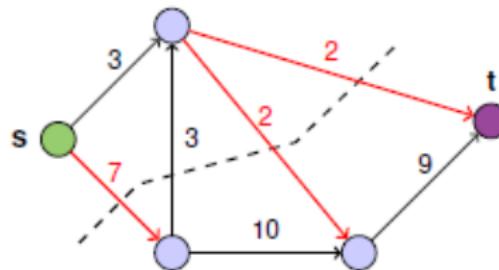
Questo ci dà un flusso totale pari a 13. Come possiamo sapere di non poter fare di meglio? ad esempio, ci sono ancora condotte non completamente utilizzate in partenza dalla sorgente s e anche tra quelle in arrivo al terminale t : chi mi garantisce che non possa arrivare ad avere un flusso totale pari a 14?

Reti di flusso

Una **sezione** di un grafo è una partizione dell'insieme dei vertici $V = P \cup \hat{P}$ tale che P contiene la sorgente s , \hat{P} contiene il terminale t .

La **capacità di una sezione** è la somma delle capacità dei lati che uniscono i punti a monte con i punti a valle della sezione:

$$k(P, \hat{P}) = \sum_{(u,v) \in E | u \in P, v \in \hat{P}} c(u, v)$$



La sezione in figura ha capacità pari a $7 + 2 + 2 = 11$

È abbastanza evidente che il flusso totale di un grafo può essere calcolato anche analizzando il flusso sui lati che attraversano una sezione:

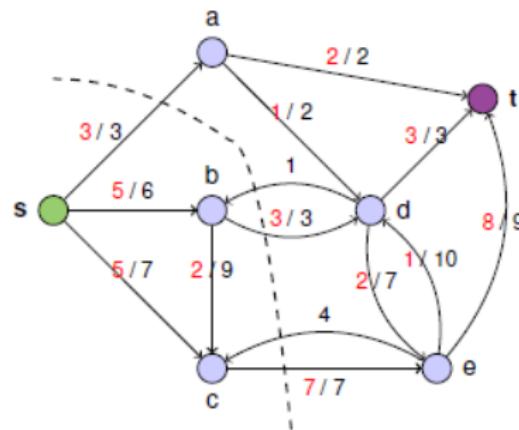
$$F(G) = \sum_{(u,v) \in E | u \in P, v \in \hat{P}} f(u, v) - \sum_{(v,u) \in E | v \in \hat{P}, u \in P} f(v, u)$$

Siccome questo è vero per ogni sezione, otteniamo che: Il flusso totale in un grafo è limitato dalla capacità di una sezione minima, ossia $F(G) \leq k(P_m, \hat{P}_m) = \min_k(P, \hat{P})$. Possiamo raggiungere questo limite? In effetti, sì. Lo garantisce questo teorema (che non dimostriamo, provate a casa!).

Teorema

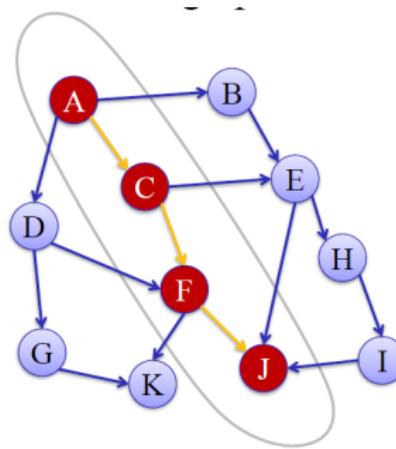
Il flusso totale in un grafo è uguale alla capacità di una sezione minimale, ossia $F(G) = k(P_m, \hat{P}_m) = \min(k(P, \hat{P}))$

Ritorniamo al nostro esempio. Abbiamo costruito un flusso ammissibile sul grafo che ha flusso totale 13.



Si prenda la sezione $\{s, b, c\} \cup \{a, d, e, t\}$: la sua capacità vale 13. Allora il flusso che abbiamo costruito è massimale.

Dato un grafo ed un nodo di partenza, selezioniamo in modo **stocastico (random)** un vicino e ci muoviamo al vicino successivo.



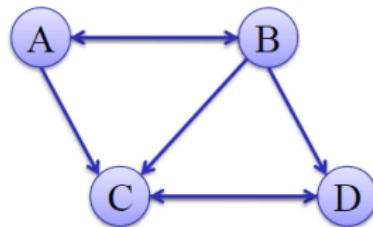
La sequenza di nodi ottenuta tramite tale visita random è detta **random walk**.

Matrice di transizione

Una **matrice di transizione** è una matrice stocastica dove ogni elemento a_{ij} rappresenta la **probabilità** di muoversi da i a j .
Ogni riga ha, quindi, **somma 1**.

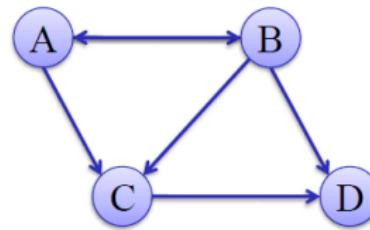
Adjacency Matrix

$$\begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$



Transition Matrix

$$\begin{bmatrix} 0 & 1/2 & 1/2 & 0 \\ 1/3 & 0 & 1/3 & 1/3 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$



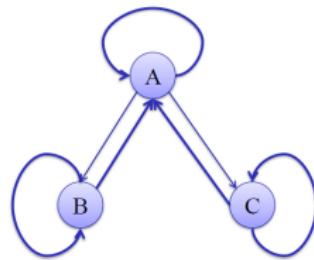
Una **catena di Markov** descrive un processo stocastico a tempo discreto su un insieme di stati $S = \{s_1, s_2, \dots, s_n\}$ in accordo con una matrice di transizione probabilistica $P = \{P_{ij}\}$, dove P_{ij} è la probabilità di muoversi dallo stato i allo stato j .

Le catene di Markov "standard" sono memory-less, quindi il prossimo stato della catena non dipende dagli stati precedenti ma solo dallo stato corrente.

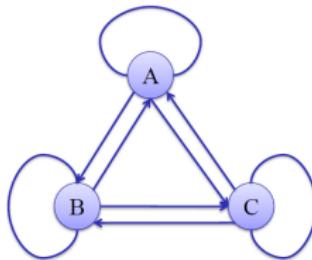
I random walk sui grafi corrispondono a catene di Markov:

- l'insieme degli stati S è l'insieme dei nodi del grafo
- la matrice di transizione corrisponde alle probabilità di seguire un determinato arco da un nodo ad un altro

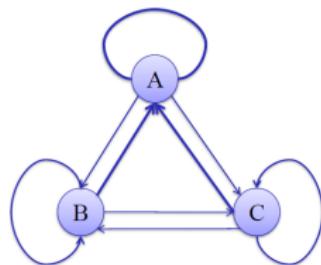
P_{ij}^k è la probabilità che il random walk partendo dal nodo i arrivi al nodo j dopo k passi.



$$p^1 = \begin{bmatrix} 0.5 & 0.25 & 0.25 \\ 0.5 & 0.5 & 0 \\ 0.5 & 0 & 0.5 \end{bmatrix}$$



$$p^2 = \begin{bmatrix} 0.5 & 0.25 & 0.25 \\ 0.5 & 0.375 & 0.125 \\ 0.25 & 0.125 & 0.375 \end{bmatrix}$$



$$p^3 = \begin{bmatrix} 0.5 & 0.25 & 0.25 \\ 0.5 & 0.3125 & 0.1875 \\ 0.5 & 0.1875 & 0.3125 \end{bmatrix}$$

Sia $x_t(i)$ la probabilità che un camminatore (testa attuale della passeggiata) sia al nodo i al tempo t , allora:

- $x_{t+1}(j) = \sum_i x_t(i) \cdot P_{ij}$
- $x_{t+1} = x_t \cdot P = x_{t-1} \cdot P \cdot P = x_{t-2} \cdot P \cdot P \cdot P = x_0 \cdot P^t$

Se il camminatore va in giro per molto tempo, alla fine si ottiene una **distribuzione stazionaria**.

Tale distribuzione per un dato nodo è associata all'ammontare delle volte che un camminatore random ha visitato il nodo in questione.

Per tempi veramente lunghi, la distribuzione tende a stabilizzarsi:
 $x_{t+1}(i) = x_t(i)$.

Inoltre, per particolari tipologie di grafi (irriducibili e aperiodici), tale distribuzione non dipende dalla distribuzione di partenza.

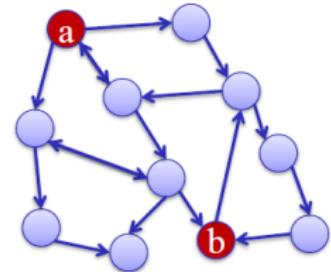
Hitting time e commute time

Lo **hitting time** dal nodo i al nodo j rappresenta il tempo (numero atteso di salti/hop) che serve per arrivare a j partendo da i :

$$h(i, j) = 1 + \sum_{k \in \text{adj}(i)} P(i, k) \cdot h(k, j)$$

Il **commute time** invece misura il tempo di andata e ritorno dai i verso j , ed è definito come

$$c(i, j) = h(i, j) + h(j, i)$$



h non è simmetrica, mentre c lo è.

Ranking delle pagine web e Pagerank

Problema: data una query e un alto numero di pagine web rilevanti per tale query, ordinare le pagine web per rilevanza secondo la topologia dettata dai collegamenti ipertestuali (hyperlink).

Pagerank (1999)

- Si simuli un random walker sul grafo del Web aventi due tipi di nodi: pagine web e argomenti in esse discussi
- Il walker salta in modo arbitrario da una pagina all'altra secondo delle probabilità non nulle
- Una pagina è importante se un'altra pagina punta ad essa

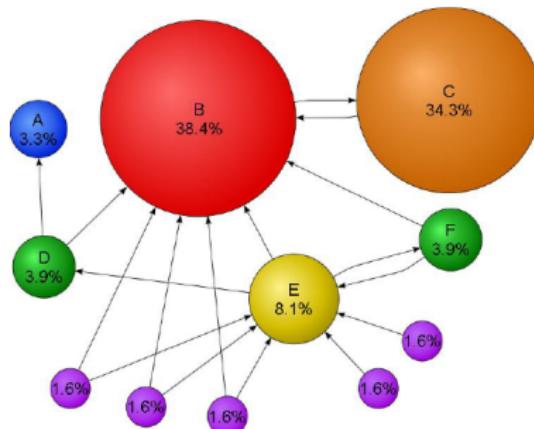
$$s(i) = \sum_{j \in adj(i)} \frac{s(j)}{deg(j)}$$

- s viene utilizzata come distribuzione stazionaria del random walk sul grafo del Web

Power Iteration

Per calcolare la distruzione stazionaria utilizziamo l'algoritmo di **Power Iteration**:

- Parti con una distribuzione x_0
- Sia $x_{t+1} = x_t \cdot P$
- Itera fino a che x_{t+1} non cambia (relativamente di molto) rispetto a x_t



Una volta terminato utilizziamo h o c per determinare il grado di similarità/rilevanza tra un argomento ed una pagina web.

Un grafo è **irriducibile** se esiste un percorso orientato tra ogni coppia di nodi.

Un grafo è **aperiodico** se il massimo comune divisore tra le lunghezze di tutti i cicli presenti nel grafo è 1.

Se una catena di Markov è un grafo irriducibile aperiodico, allora l'autovalore più grande della matrice di transizione è uguale a 1 e tutti gli altri autovalori saranno prossimi a 1.

Se un grafo non è irriducibile o aperiodico, possiamo ovviare a tale problema utilizzando una probabilità di restar c .

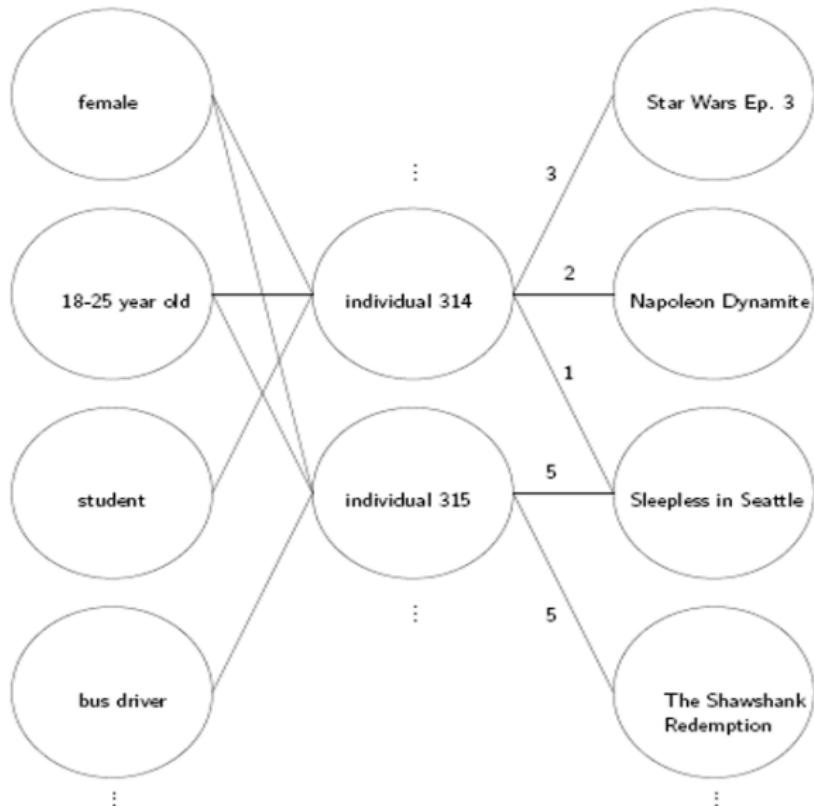
Ad ogni iterazione il walker può catapultarsi in un nodo del grafo qualsiasi con probabilità c .

Inoltre, il walker salta sui vicini diretti del nodo corrente con probabilità $1 - c$.

Otteniamo così una nuova matrice $\tilde{P} = (1 - c) \cdot P + c \cdot U$, con $U_{ij} = \frac{1}{n} \forall i, j$.

U è una distruzione uniforme. Tuttavia, possiamo utilizzare una qualsiasi distribuzione di probabilità per teletrasportarsi da un nodo all'altro, ottenendo così **Personalized Pagerank**, dando un vettore di preferenza per le specifiche pagine web = utenti di un sistema di raccomandazione.

Random walk con restart



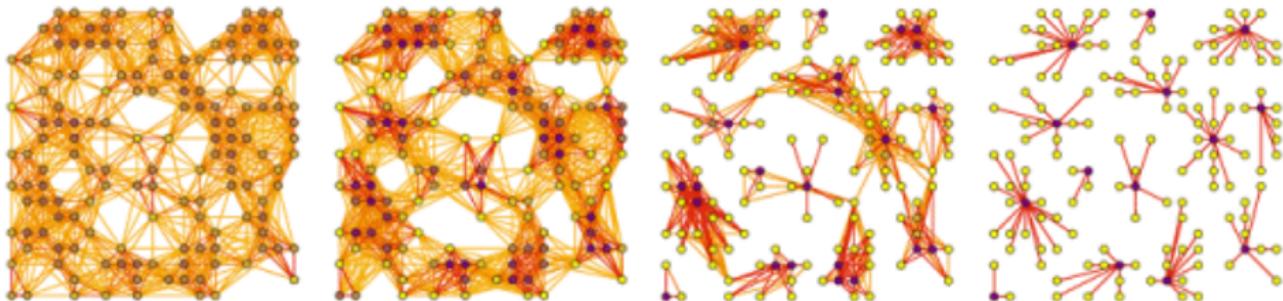
MCL- Markov CLustering (van Dongen 2000)

MCL è un algoritmo di **clustering** basato sui **grafi**.

Ingredienti:

- Distanza di tempo di percorrenza (**commute time**) euclidea
- Un misura di distanza basata sui **random walk** in un grafo
- **K-means** su tale misura di distanza

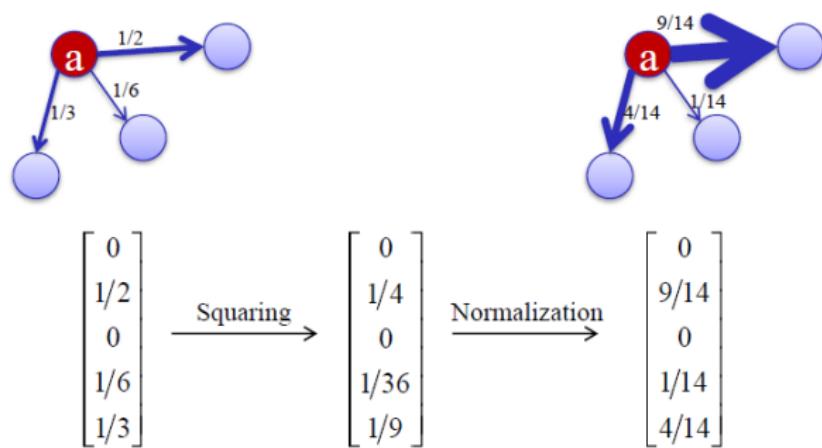
L'**idea** è che tra i nodi dello stesso cluster vi siano molti archi e che vi siano meno archi tra i nodi di cluster diversi. quindi, un random walk con restart che parte da un nodo i ha maggiore probabilità di restare nel cluster a cui i appartiene.

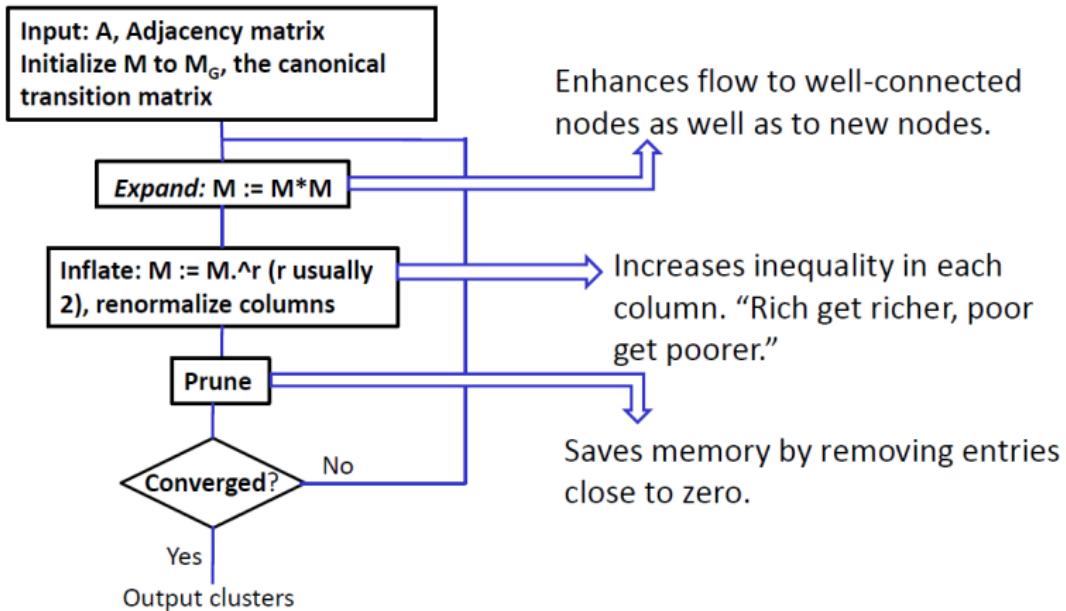


Il **flusso stocastico** è più facile dentro un cluster che tra due cluster.
Per simulare tale flusso eleviamo la matrice di transizione a potenze intere,
ovvero ad ogni passo di iterazione del random walk facciamo una
moltiplicazione tra matrici.

Problema: i pesi associati agli archi tra nodi dello stesso cluster saranno
forti nelle iterazioni iniziali ma andranno via via ad indebolirsi.

MCL, quindi, corre ai ripari
fermando le iterazioni e
aggiustando i pesi con un
processo di **inflazione** tale
che i pesi tra archi dello
stesso cluster sono rafforzati
ed i pesi di archi tra cluster
sono indeboliti.

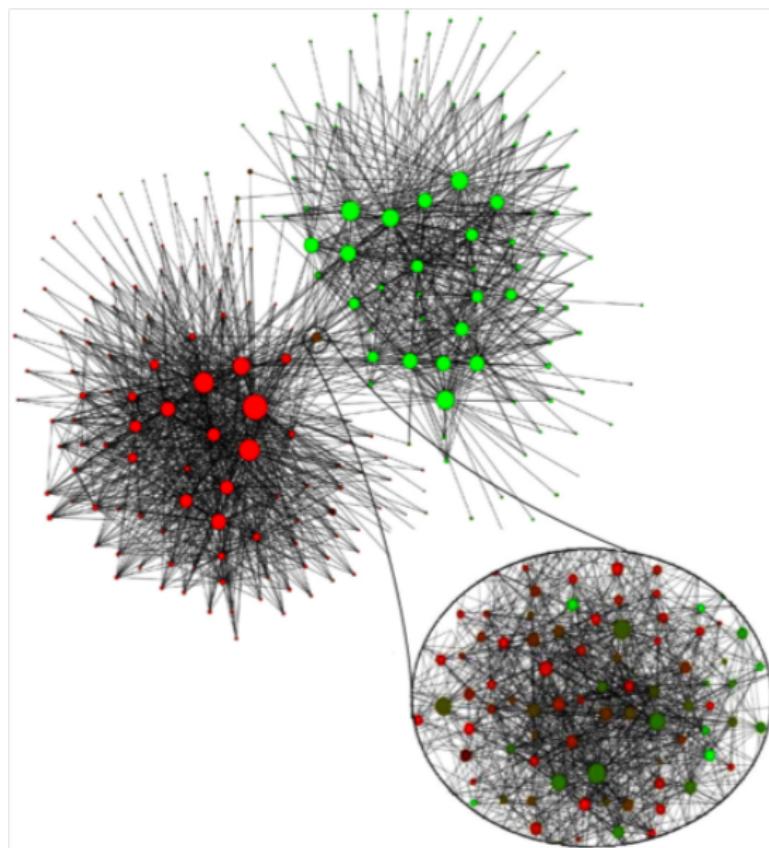




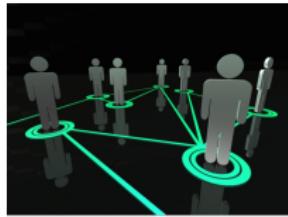
Il Belgio è spesso visto come il prototipo di una società che ha trovato un modo per far coesistere pacificamente due culture per secoli, senza che una cultura assorbisse l'altra. In effetti, il paese è bilingue: il 59% dei suoi cittadini si definisce fiammingo e parla olandese. L'altro 40% sono valloni e parlano francese.

Nel 2008 Vincent Blondel e i suoi studenti hanno iniziato ad applicare un nuovo algoritmo di ricerca della comunità ai modelli di chiamata di uno dei più grandi operatori di telefonia mobile in Belgio. È stato progettato per identificare gruppi di individui che parlano regolarmente tra loro al telefono, suddividendo l'intero paese in numerose comunità piccole e non così piccole, mettendo gli individui accanto ai loro amici, familiari, colleghi, vicini, tutti quelli che chiamavano regolarmente sul loro cellulare. Il risultato è stato in qualche modo inaspettato: ha indicato che il Belgio è diviso in due grandi comunità, ciascuna composta da molte cerchie più piccole di amici. All'interno di ciascuno di questi due gruppi, le comunità avevano molteplici legami tra loro. Eppure, queste comunità non hanno mai parlato con le comunità dell'altro gruppo. Tra questi due mega-gruppi c'era un terzo gruppo, molto più piccolo, di comunità, che apparentemente mediavano tra le due parti del Belgio.

Community detection



Community detection: scala mesoscopica



Microscopic



Mesoscopic

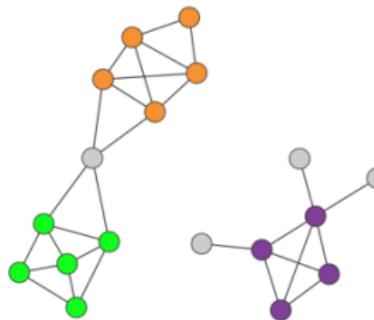
Macroscopic



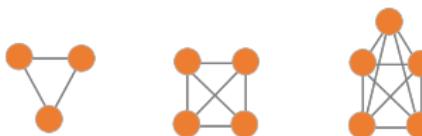
H1: La struttura della comunità di una rete è codificata in modo univoco nella sua topologia (wiring diagram).

H2 - ipotesi di connettività: una comunità corrisponde ad un sottografo connesso

H2 - ipotesi di densità: una comunità corrisponde ad un vicinato localmente denso.

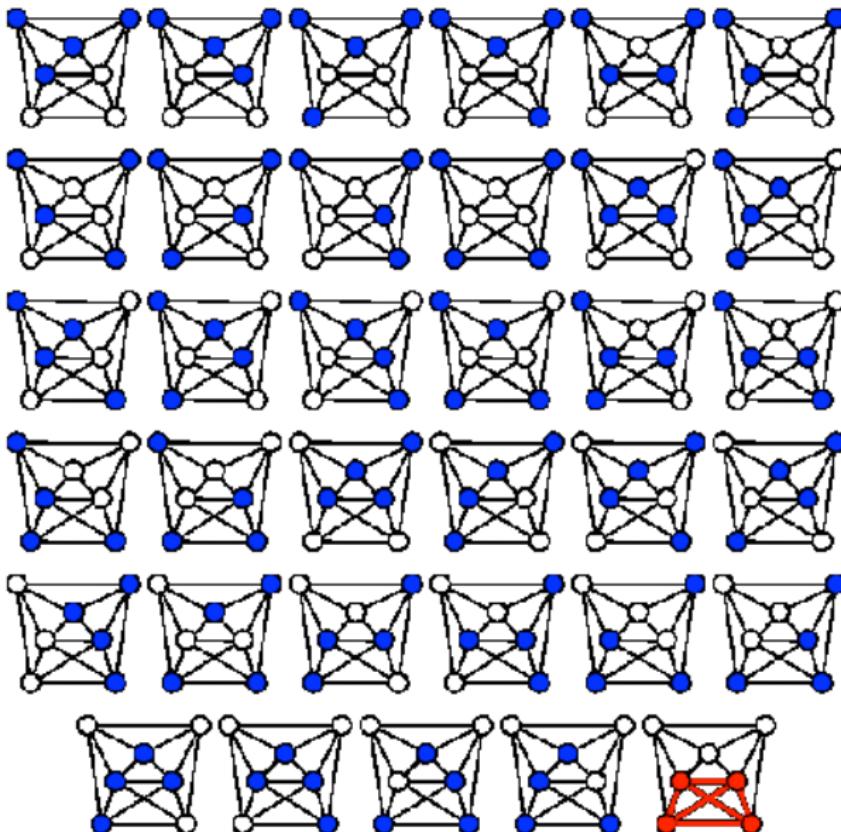


Una k-cricca (k-clique) è un grafo completo di k nodi.



- I triangoli sono frequenti; Le cricche più grandi sono rare.
- Le comunità non corrispondono necessariamente a sottografi completi, poiché molti dei loro nodi non si collegano direttamente tra loro.
- Trovare le cricche di una rete è computazionalmente piuttosto impegnativo, essendo un cosiddetto problema NP-completo

Clique come comunità



Si consideri un sottografo C connesso di nodi N_C .

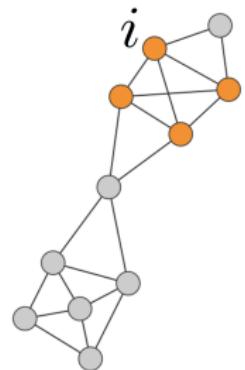
Grado interno, k_i^{int} : insieme di collegamenti del nodo i che si collega ad altri nodi della stessa comunità C .

$$k_i^{int} = 3$$
$$k_i^{ext} = 1$$

Grado esterno k_i^{ext} : l'insieme dei collegamenti del nodo i che si connette al resto della rete.

Se $k_i^{ext} = 0$: tutti i vicini di i appartengono a C , e C è una buona comunità per i .

Se $k_i^{int} = 0$, tutti i vicini di i appartengono ad altre comunità, allora dovrei essere assegnato a una comunità diversa.



Comunità forti e deboli

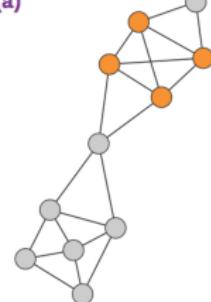
Comunità forte: Ogni nodo di C ha più collegamenti all'interno della comunità che con il resto del grafo

$$k_i^{int}(C) > k_i^{ext}(C)$$

Comunità debole: Il grado interno totale di C supera il suo grado esterno totale

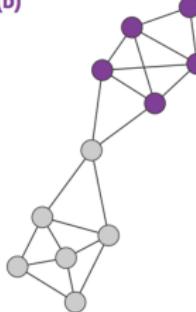
$$\sum_{i \in C} k_i^{int}(C) > \sum_{i \in C} k_i^{ext}(C)$$

(a)



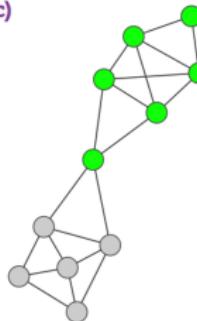
Clique

(b)



Strong

(c)



Weak

In quanti modi possiamo suddividere una rete in 2 comunità?

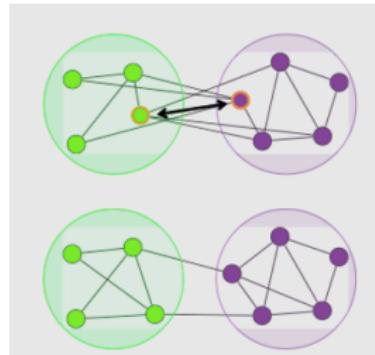
Bisezione di grafi: dividi una rete in due sottografi uguali non sovrapposti, in modo tale che il numero di collegamenti tra i nodi nei due gruppi sia ridotto al minimo.

Si formano due sottogruppi di dimensione n_1 e n_2 . Il numero totale di combinazioni è $\frac{N!}{n_1!n_2!}$.

Il partizionamento (bisezione) dei grafi mira a dividere una rete in un numero predefinito di sottografi più piccoli. Al contrario, il rilevamento della comunità mira a scoprire la struttura intrinseca della comunità che caratterizza una rete. Di conseguenza, nel rilevamento delle comunità il numero e la dimensione delle comunità non sono predefiniti, ma sono codificati nello schema elettrico della rete e devono essere scoperti.

Algoritmo di bisezione Kernighan-Lin

- Partizionare una rete in due gruppi di dimensioni predefinite. Questa partizione è chiamata **taglio**.
- Ispeziona ciascuno una coppia di nodi, uno per ogni gruppo. Identificare la coppia che si traduce nella maggiore riduzione della dimensione del taglio (collegamenti tra i due gruppi) se li scambiamo
- Scambiali.
- Se nessuna coppia riduce la dimensione del taglio, scambiamo la coppia che aumenta meno la dimensione del taglio.
- Il processo viene ripetuto fino a quando ogni nodo non viene spostato una volta.

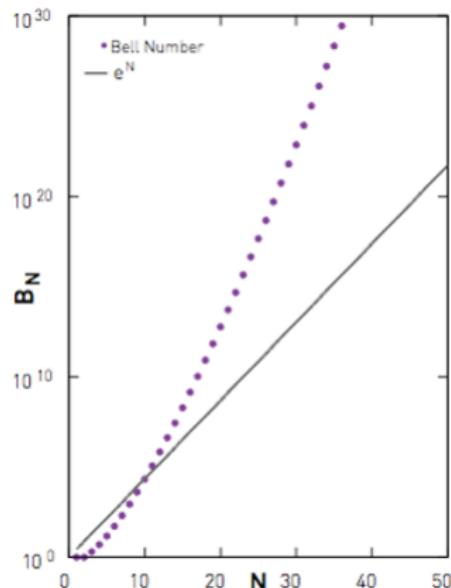


Rilevamento della comunità: Il numero e la dimensione delle comunità sono sconosciuti all'inizio.

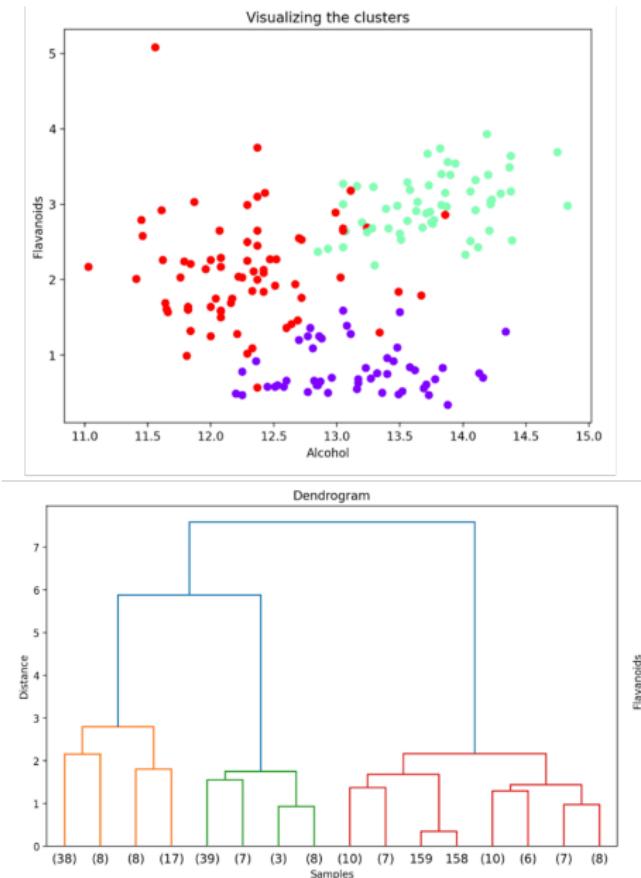
Partizione: Divisione di una rete in gruppi di nodi, in modo che ogni nodo appartenga a un gruppo.

$$B_N = \frac{1}{e} \sum_{j=0}^{\infty} \frac{j^N}{j!}$$

con B_N l' N -esimo numero di Bell.

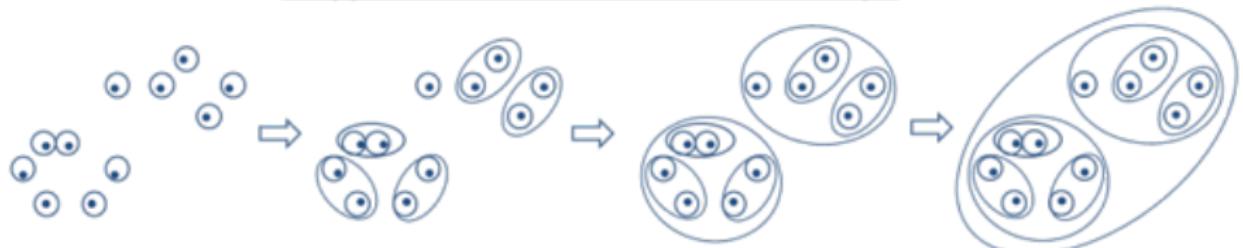


Clustering gerarchico

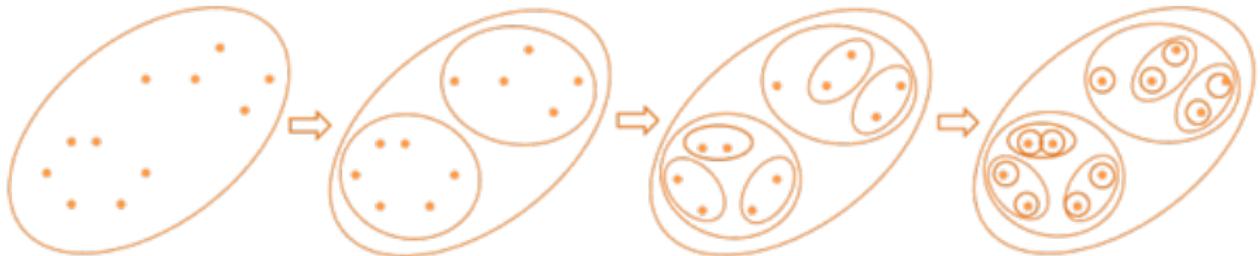


Clustering gerarchico

Agglomerative Hierarchical Clustering



Divisive Hierarchical Clustering

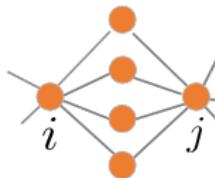




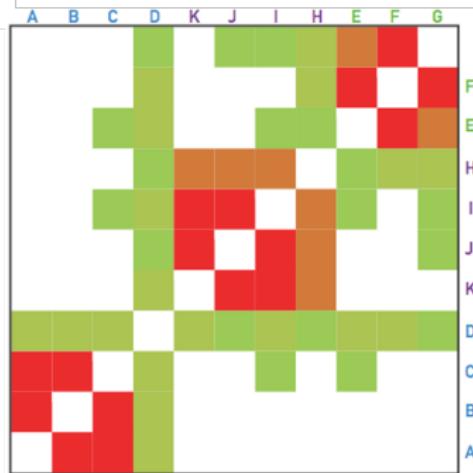
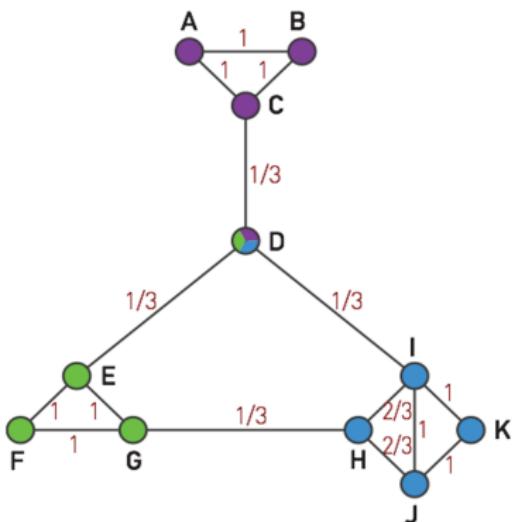
Gli algoritmi agglomerativi uniscono nodi e comunità con un'elevata somiglianza.

Passo 1: definire la matrice di somiglianza (algoritmo di Ravasz)

- Alto per le coppie di nodi che probabilmente appartengono alla stessa comunità, basso per quelli che probabilmente appartengono a comunità diverse.
- I nodi che si connettono direttamente tra loro e/o condividono più vicini hanno maggiori probabilità di appartenere allo stesso denso vicinato locale, quindi la loro somiglianza dovrebbe essere grande.
Matrice di sovrapposizione topologica: $X_{ij}^o = \frac{J_n(i,j)}{\min(k_i, k_j)}$
 $J(i,j)$: numero di vicini in comune tra i nodi i e j ; +1 se gli archi sono diretti

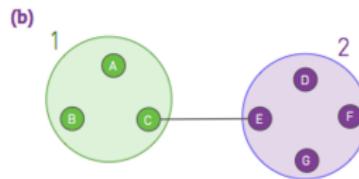
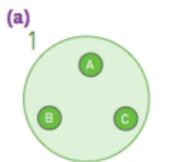


Algoritmi agglomerativi

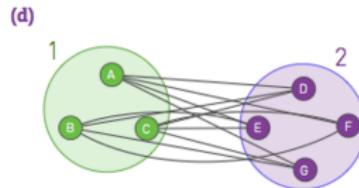
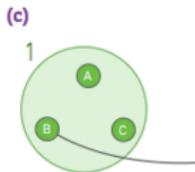


Passo 2: decidi la somiglianza del gruppo

I gruppi vengono uniti in base alla loro somiglianza reciproca attraverso il collegamento a cluster singolo, completo o medio.



$$\frac{1}{x_{ij}} = r_{ij} = \begin{array}{c|cccc} & D & E & F & G \\ \hline A & 2.75 & 2.22 & 3.46 & 3.08 \\ B & 3.38 & 2.68 & 3.97 & 3.40 \\ C & 2.31 & 1.59 & 2.88 & 2.34 \end{array}$$



Passo 3: Applicare il clustering gerarchico.

- Assegna ogni nodo a una comunità a sé stante e valuta la somiglianza per tutte le coppie di nodi. Le somiglianze iniziali tra queste "comunità" sono semplicemente le somiglianze dei nodi.
- Trova la coppia di comunità con la più alta somiglianza e uniscile per formare un'unica comunità.
- Calcola la somiglianza tra la nuova comunità e tutte le altre comunità.
- Ripetere dal passaggio 2 fino a quando tutti i nodi non vengono uniti in un'unica comunità.

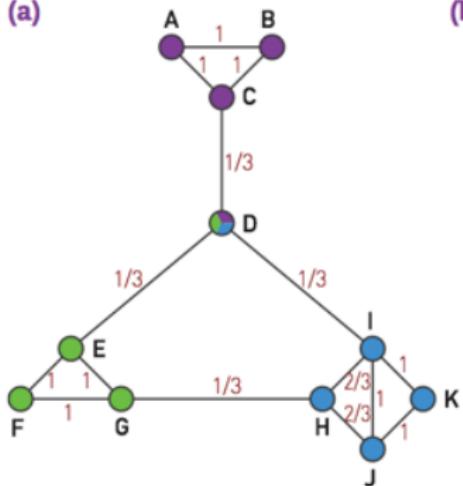
Passo 4: costruisci il dendrogramma

- Viene descritto l'ordine preciso in cui i nodi vengono assegnati alle comunità.

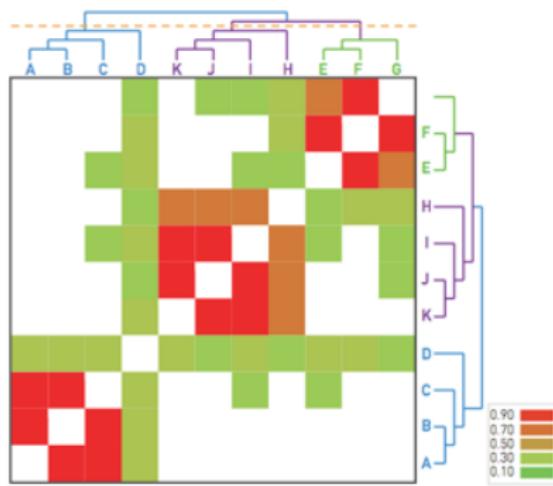
Complessità $O(N^2)$.

Algoritmi agglomerativi

(a)

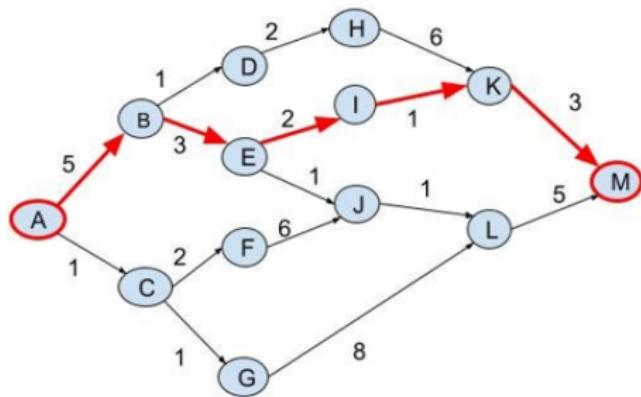


(b)

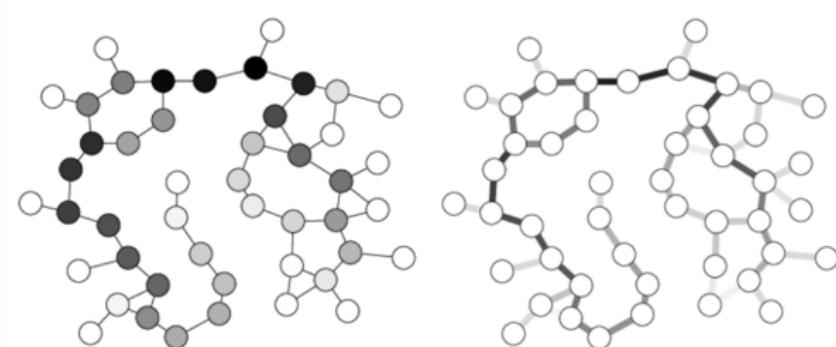


Dati due nodi, A e M, il percorso corto che li collega è il percorso da A a M avente il numero minimo di spigoli (nodi).

Per le reti ponderate, dobbiamo tenere conto dei pesi. Quindi, il percorso corto è il percorso che ha la somma minima dei suoi spigoli tra tutti i percorsi che collegano A a M.



Dato un nodo (edge), la sua betweenness è data dal numero di percorsi più brevi (tra una qualsiasi coppia di nodi della rete) che passano attraverso di esso.



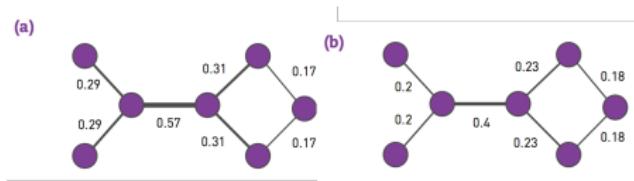
(a) Node betweenness.

(b) Edge betweenness.

Gli algoritmi divisivi dividono le comunità rimuovendo i collegamenti che collegano i nodi con bassa somiglianza.

Passo 1: definire una misura di centralità (algoritmo di Girvan-Newman).

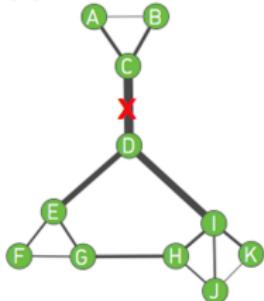
- La betwenness è il numero di percorsi più brevi tra tutte le coppie di nodi che corrono lungo un collegamento.
- Random-wlak betwenneess: Una coppia di nodi m e n viene scelta a caso. Un camminatore inizia da m, seguendo ogni collegamento adiacente con uguale probabilità fino a raggiungere n. La camminata casuale tra x_{ij} è la probabilità che il collegamento $i \rightarrow j$ sia stato attraversato dal camminatore dopo aver fatto la media di tutte le possibili scelte per i nodi di partenza m e n



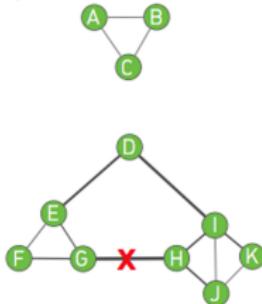
Passo 2: Clustering gerarchico

- Calcolo della centralità di ogni collegamento.
- Eliminare il collegamento con la centralità maggiore; In caso di pareggio, sceglie uno a caso.
- Ricalcola la centralità di ogni collegamento per la rete modificata.
- Ripetere fino a quando tutti i collegamenti vengono rimossi (si ottiene un dendrogramma).

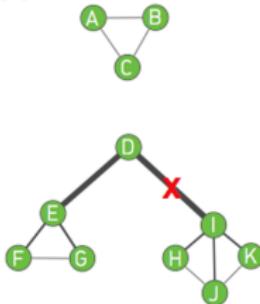
(a)



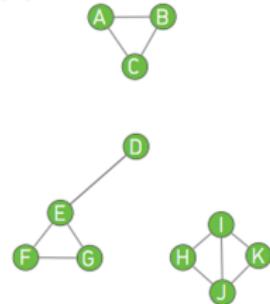
(b)



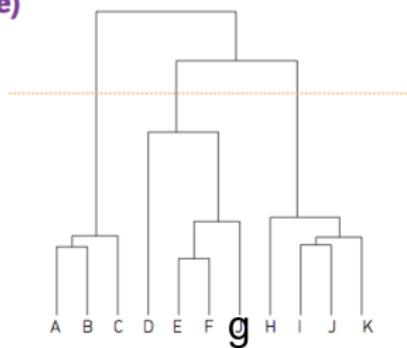
(c)



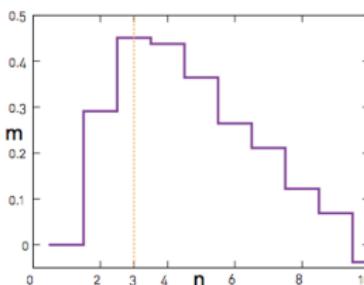
(d)



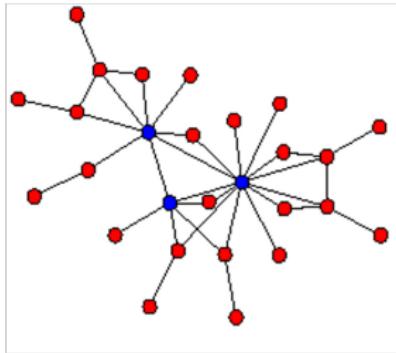
(e)



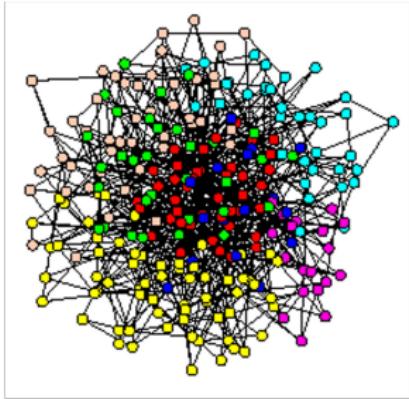
(f)



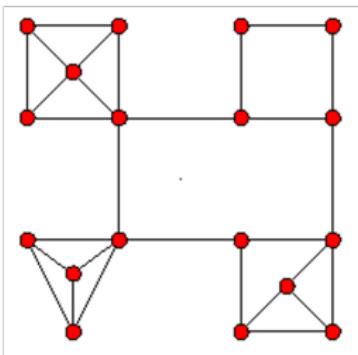
(a)



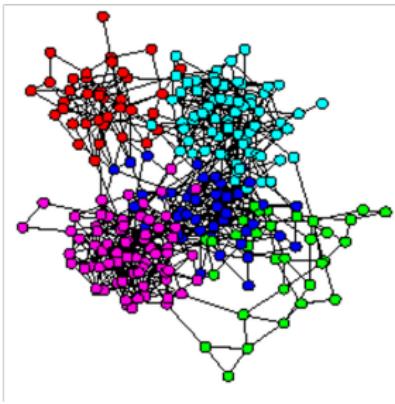
Scale-free



(b)

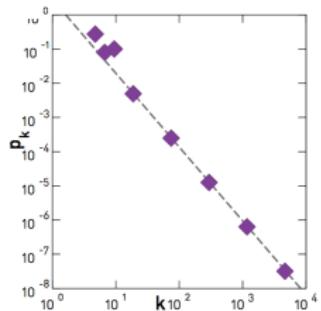


Modular



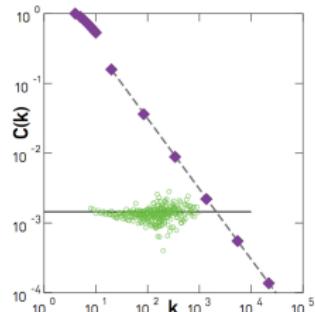
1. Scale-free

$$\gamma = 1 + \frac{\ln 5}{\ln 4} = 2.161$$



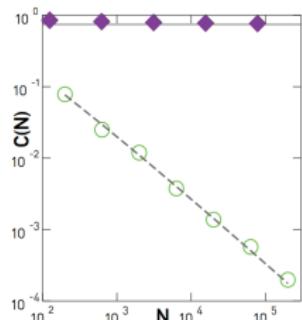
2. Scaling clustering coefficient (DGM)

$$C(k) \sim k^{-1}$$

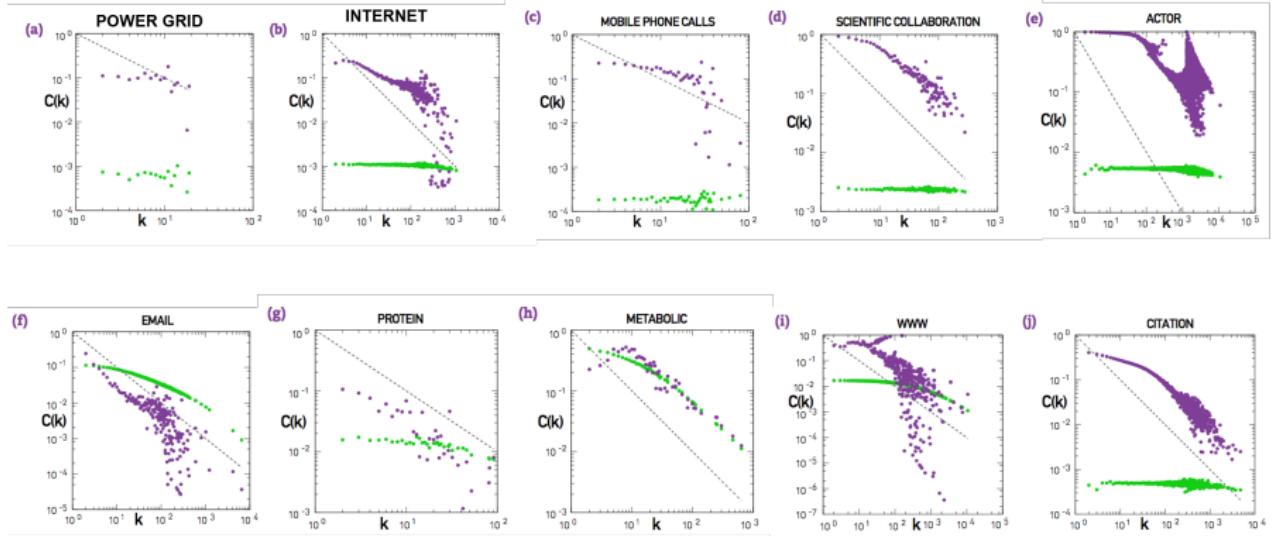


3. Clustering coefficient independent of N

$$C(N) = \text{const.}$$



Gerarchia delle reti



Nelle reti casuali (random) ci si aspetta che il modello di connessione sia uniforme, privo di fluttuazioni di densità locale che potremmo interpretare come comunità.

H4: ipotesi di randomicità: reti casuali non dovrebbero avere strutture comunitarie.

Sia data una partizione di n_c comunità $\{C_c, c = 1, n_c\}$

$$\text{Modularità} = M(C_c) = \frac{1}{2|E|} \sum_{i,j \in C_c}^N (A_{ij} - P_{ij}) = \frac{|E_c|}{|E|} - \left(\frac{k_c}{2|E|}\right)^2$$

- A_{ij} il dato reale
- P_{ij} la probabilità di avere quell'arco rispetto al modello considerato.
Nelle reti random $P_{ij} = 2|E|p_i p_j = \frac{k_i k_j}{2|E|}$
- E_c gli archi all'interno della comunità
- k_c il grado totale dei nodi della comunità

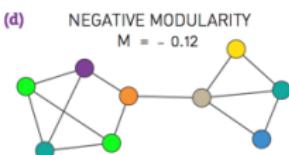
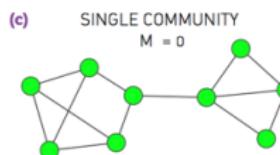
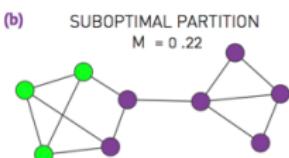
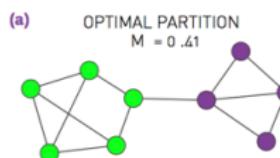
Da cui:

$$M = \sum_{c=1}^{n_c} \frac{|E_c|}{|E|} - \left(\frac{k_c}{2|E|}\right)^2$$

H5 - ipotesi di massima modularità

La partizione con la massima modularità M per una data rete offre la struttura ottimale della comunità.

L'obiettivo è trovare il partizionamento $\{C_c, c = 1, n_c\}$ che massimizza M .



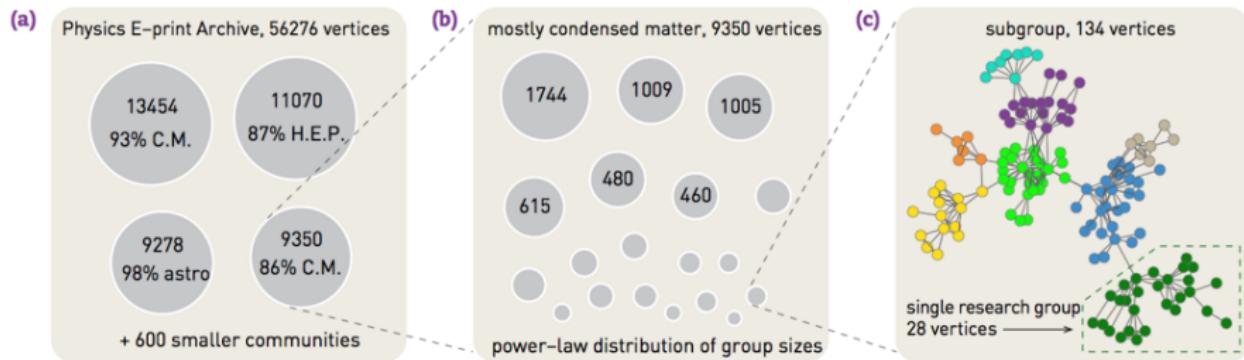
- Partizione ottimale, che massimizza la modularità.
- Modularità non ottimale ma positiva.
- Modularità negativa: Se assegniamo ogni nodo a una comunità diversa.
- Modularità zero: assegnazione di tutti i nodi alla stessa comunità, indipendentemente dalla struttura della rete.

La modularità dipende dalle dimensioni

Un algoritmo greedy, che unisce iterativamente i nodi se lo spostamento aumenta la modularità della nuova partizione.

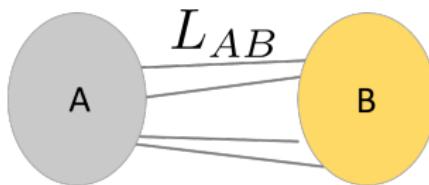
- ① Assegna ogni nodo a una comunità a sé stante. Quindi iniziamo con N comunità.
- ② Ispezionare ogni coppia di comunità collegate da almeno un collegamento e calcolare la variazione di modularità ottenuta se uniamo queste due comunità.
- ③ Identificare le coppie di comunità per le quali ΔM è il più grande e unirle. Si noti che la modularità di una particolare partizione viene sempre calcolata dalla topologia completa della rete.
- ④ Ripetere il passaggio 2 fino a quando tutti i nodi non vengono uniti in un'unica comunità.
- ⑤ Registra per ogni passaggio e seleziona la partizione per la quale la modularità è massima.

Community detection basata sulla modularità



Si consideri la rete di collaborazione tra fisici, composta da $N=56.276$ scienziati di tutte le branche della fisica che hanno pubblicato articoli su arxiv.org (Figura). L'algoritmo greedy prevede circa 600 comunità con una modularità di picco $M = 0,713$. Quattro di queste comunità sono molto grandi, insieme contengono il 77% di tutti i nodi (Figura a). Nella comunità più grande, il 93% degli autori pubblica in fisica della materia condensata, mentre l'87% degli autori nella seconda comunità più grande pubblica in fisica delle alte energie, indicando che ogni comunità rilevata contiene fisici con interessi professionali simili.

Limiti della modularità



siano date due comunità, A e B . L_{AB} è il numero di archi tra i nodi di A e B , ed L il numero totali di archi della rete. La differenza di modularità è data da:

$$\Delta M_{AB} = \frac{L_{AB}}{L} - \frac{k_a k_b}{2L^2}$$

con k_a e K_b rispettivamente il grado totale in A e B .

Se $\frac{k_a k_b}{2L} < 1$ e $L_{AB} \geq 1$ allora $\Delta M_{AB} > 0$ allora fondiamo le due comunità per massimizzare M .

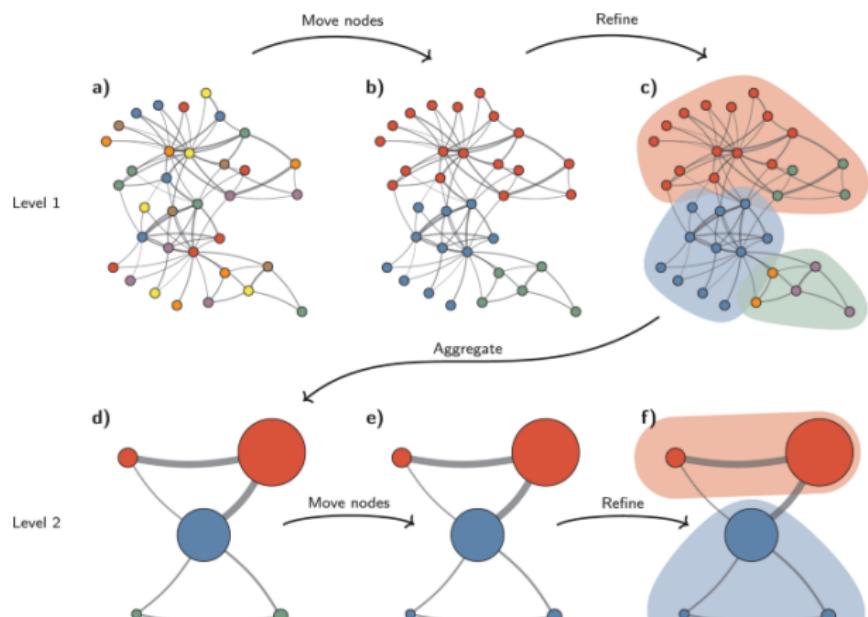
Se assumiamo $k_a \sim k_b = k$ allora $k \leq \sqrt{2L}$.

La modularità ha un limite di risoluzione. Infatti, non può trovare comunità più piccole di questa grandezza.

Questo approccio si basa sulla **modularità**, che cerca di **massimizzare** la differenza tra il numero effettivo di edge in una comunità e il numero previsto di edge nella comunità. L'ottimizzazione della modularità in una rete è NP-hard, quindi è necessario utilizzare un'euristica.

L'algoritmo di Louvain è suddiviso in due fasi ripetute in modo iterativo

- Spostamento locale dei nodi
- Aggregazione della rete



Per un grafo pesato la **modularità** è definita come:

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

- A_{ij} è il peso dell'arco dal nodo i al nodo j
- k_i e k_j sono le somme dei pesi degli archi che toccano i e j rispettivamente
- m è la somma dei pesi di tutti gli archi
- c_i e c_j sono le comunità assegnate ai due nodi; $\delta(c_i, c_k)$ è la funzione di Kronecker tale che $\delta(x, y) = 1$ se $x = y$, 0 altrimenti.

Louvain - spostamento locale dei nodi

Per una determinata comunità c , può essere calcolato come:

$$Q_c = \frac{\sum_{in}}{2m} = \left(\frac{\sum_{tot}}{2m} \right)^2$$

dove

- \sum_{in} è la somma dei pesi degli archi all'interno della comunità c
- \sum_{tot} è la sommatoria dei pesi degli archi di tutto il grafo.

Ogni nodo è inizialmente identificato come una singola comunità, quindi viene fatto partire il processo iterativo.

Per ogni nodo i , viene calcolata la modifica della modularità rimuovendo i dalla propria comunità e spostandolo nella comunità di ciascun vicino j di i :

$$\Delta Q = \left[\frac{\sum_{in} + 2k_{i,in}}{2m} - \left(\frac{\sum_{tot} + k_i}{2m} \right)^2 \right] - \left[\frac{\sum_{in}}{sm} - \left(\frac{\sum_{tot}}{2m} \right)^2 - \left(\frac{k_i}{2m} \right)^2 \right]$$

tale che $k_{i,in}$ è la somma dei pesi degli archi tra i e tutti i nodi della comunità in cui si sta spostando i .

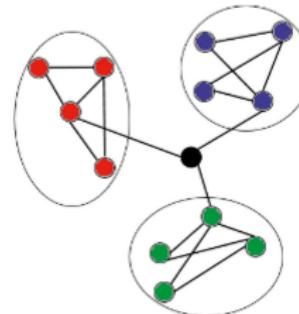
Louvain - spostamento locale dei nodi

Quindi, una volta calcolato questo valore per tutte le comunità a cui è connesso i , i viene inserito nella comunità che ha determinato il maggiore aumento della modularità.

Se nessun aumento è possibile, i rimane nella sua comunità originale.

Questo processo viene applicato ripetutamente e in sequenza a tutti i nodi fino a quando non può verificarsi alcun aumento della modularità.

Una volta raggiunto questo massimo locale di modularità, la prima fase è terminata.



La seconda fase dell'algoritmo prevede la riduzione delle comunità a un singolo nodo e la ripetizione dei passaggi della Fase 1:

- ① Ogni comunità C_j è ridotta ad un unico nodo. Gli archi che collegano i nodi da C_j al altre comunità sono similmente ridotte ad un unico arco pesato.
- ② Una volta creato il nuovo grafo, la seconda fase è terminata e la prima fase può essere riapplicata alla nuova rete.

A causa dei limiti della modularità, è stata introdotta una misura basata sulle probabilità classiche nota come Surprise per valutare la qualità di una partizione di una rete in comunità.

L'algoritmo è quasi simile all'algoritmo di rilevamento della comunità di Louvain, tranne per il fatto che utilizza sorprese invece della modularità.

I nodi vengono spostati da una comunità all'altra in modo tale da migliorare le sorprese in modo greedy.

Questo approccio considera la probabilità che un collegamento si trovi all'interno di una comunità.

L'uso delle sorprese funziona bene per trovare comunità molte piccole e l'uso della modularità funziona bene per scovare grandi comunità.

Il modello di community di Surprise è dato da

$$Q = mD(q||<q>)$$

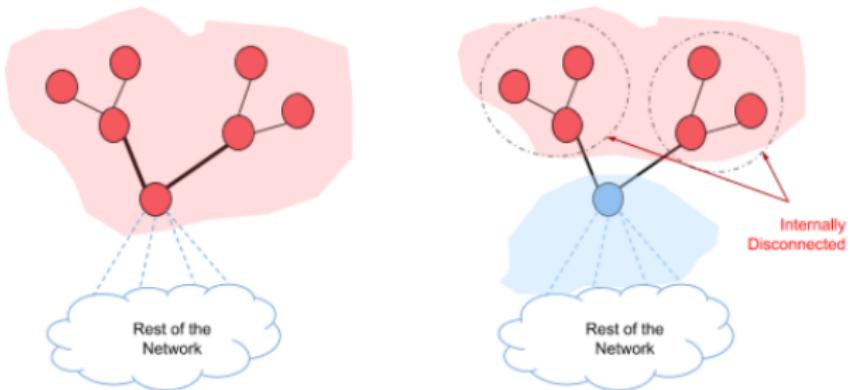
- m è il numero di archi
- $q = \frac{\sum_c m_c}{m}$ è la frazione di archi interni
- $<q> = \frac{\sum_c \binom{n_c}{2}}{\binom{n}{2}}$ è la frazione interna di archi attesa
- $D(x||y) = x\ln\frac{x}{y} + (1-x)\ln\frac{1-x}{1-y}$ è la divergenza entropica

Il rilevamento della comunità di Louvain ha la tendenza a scoprire **comunità disconnesse** internamente (comunità mal collegate).

Nell'algoritmo di Louvain, lo spostamento di un nodo che ha agito da **ponte** tra due componenti di una comunità in una nuova comunità può disconnettere la vecchia comunità.

Questo non sarà un problema se la vecchia comunità verrà ulteriormente divisa. Ma potrebbe succedere che altri nodi nella vecchia comunità le consentono di rimanere come un'unica comunità grazie alle loro forti connessioni.

Inoltre, Louvain ha la tendenza a scoprire comunità connesse in modo molte debole. Pertanto, è stato proposto l'algoritmo di Leiden molto più veloce che garantisce che le comunità siano ben collegate.



Oltre alle fasi utilizzate nell'algoritmo di Louvian, Leiden utilizza un'altra fase che cerca di perfezionare le partizioni scoperte.

Le tre fasi nell'algoritmo di Leiden sono:

- Spostamento locale dei nodi
- Affinamento delle partizioni
- Aggregazione della rete basata su partizioni raffinate

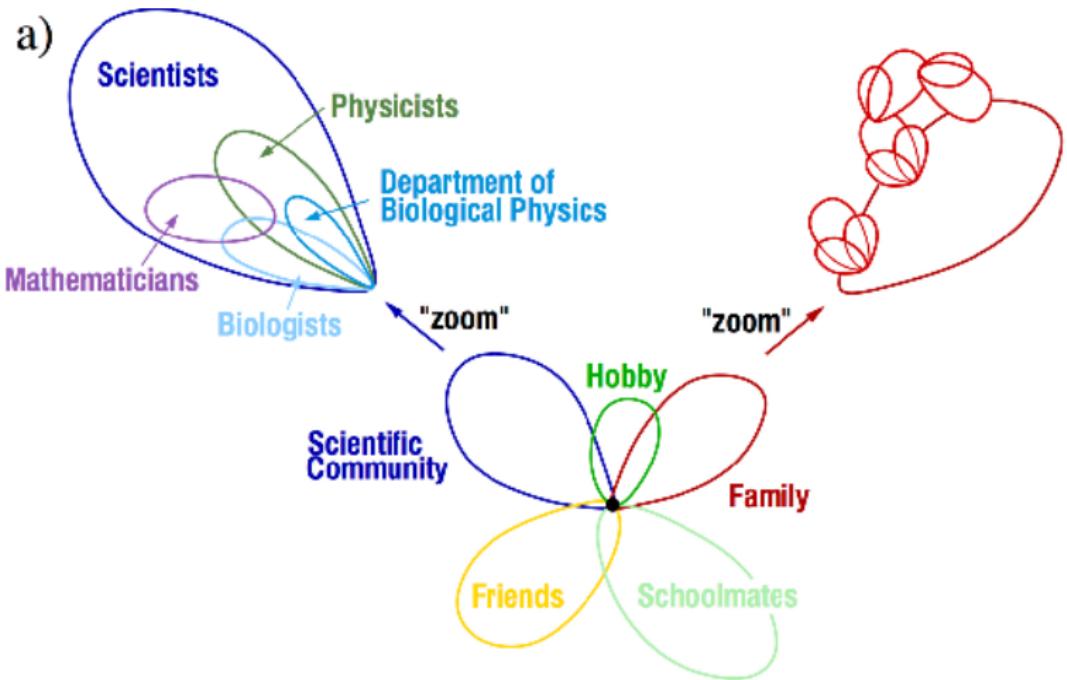
Nella fase di raffinamento, l'algoritmo cerca di identificare le partizioni raffinate dalle partizioni proposte dalla prima fase.

Le comunità proposte dalla prima fase possono ulteriormente suddividersi in più partizioni nella seconda fase.

La fase di raffinamento non segue un approccio greedy e, invece, può fondere un nodo con una comunità scelta a caso che aumenta la funzione di qualità.

Questa casualità consente di scoprire lo spazio di partizione in modo più ampio.

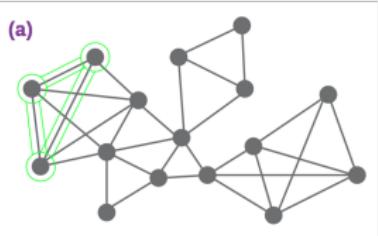
Anche nella prima fase, Leiden segue un approccio diverso al Louvain. Invece di visitare tutti i nodi della rete dopo che la prima visita a tutti i nodi è stata completata, Leiden visita solo quei nodi il cui vicinato è cambiato.



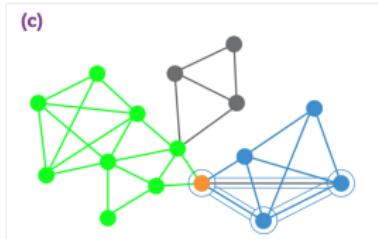
Una **k -clique** (cricca) è un grafo completo (siete un arco tra tutte le possibili coppie di nodi) di k nodi.

Due k -clique si dicono **adiacenti** se esse condividono $k - 1$ nodi.

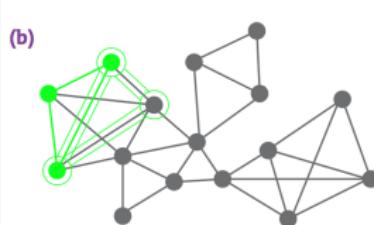
k -clicque che possono essere reciprocamente raggiunte tramite una relazione di adiacenza appartengono alla stessa comunità, altrimenti appartengono a comunità diverse.



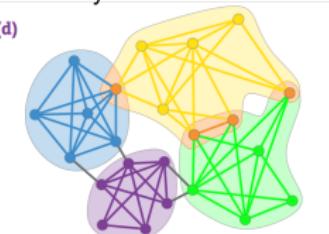
Start with a k -clique (complete subgraphs of k nodes), a 3-clique for example



A k -clique community is the largest connected subgraph obtained by the union of all adjacent k -cliques



Start “rolling” the clique over adjacent cliques. Two k -cliques are considered adjacent if they share $k-1$ nodes



Other k -cliques that can not be reached from a particular clique correspond to other clique-communities

Le comunità individuate da CFinder potrebbero emergere per caso?

Per distinguere le comunità k-clique reali dalle comunità che sono una pura conseguenza dell'elevata densità di link, esploriamo le proprietà di percolazione delle k-clique in una rete casuale/random.

La teoria della percolazione descrive il comportamento di una rete quando vengono rimossi nodi o collegamenti.

Una grande comunità di k-clique emerge in una rete casuale solo se la probabilità di connessione p supera la soglia:

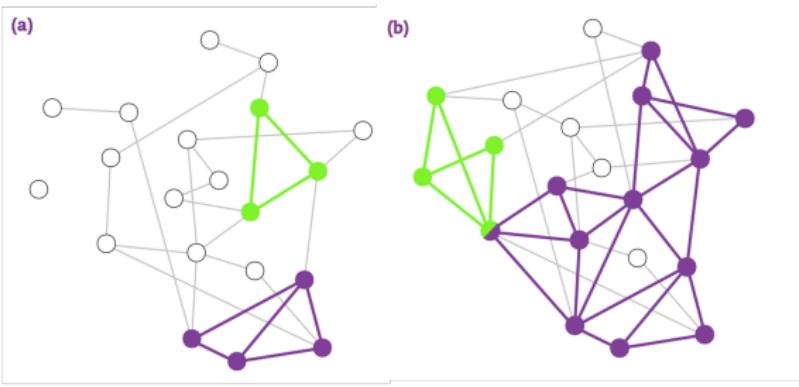
$$p_c(k) = \frac{1}{[(k-1)N]^{\frac{1}{k-1}}}$$

... in accordo con il modello Barabasi.

Percolazione - CFinder

$$k = 3 \quad p_c(k) = 1/\sqrt{2N}$$

$$N=20 \quad p_c=0.16$$



Random networks with

$$p=0.13 (< p_c)$$

$$p=0.22 (> p_c)$$

- I nodi tendono ad appartenere a più comunità
- I collegamenti tendono ad essere specifici, catturando la natura della relazione tra due nodi.

Social network, un link può indicare che due soggetti:

- appartengono alla stessa famiglia
- lavorano insieme
- Condividono un hobby.

Reti biologiche:

- Ogni interazione di una proteina è responsabile di una funzione diversa, che definisce in modo univoco il ruolo della proteina nella cellula

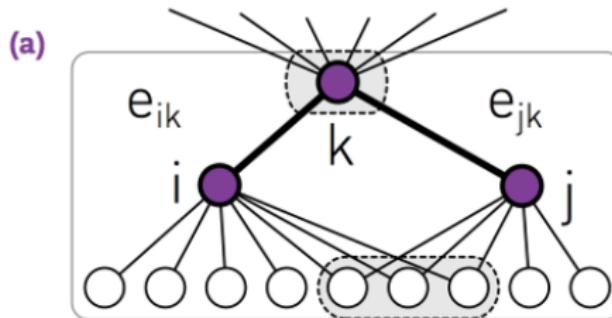
Obiettivo: definire una algoritmo gerarchico basato sulla somiglianza dei collegamenti.

Passo 1: definire la similarità

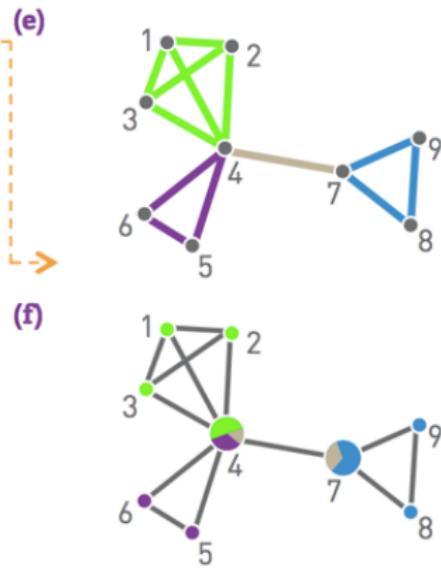
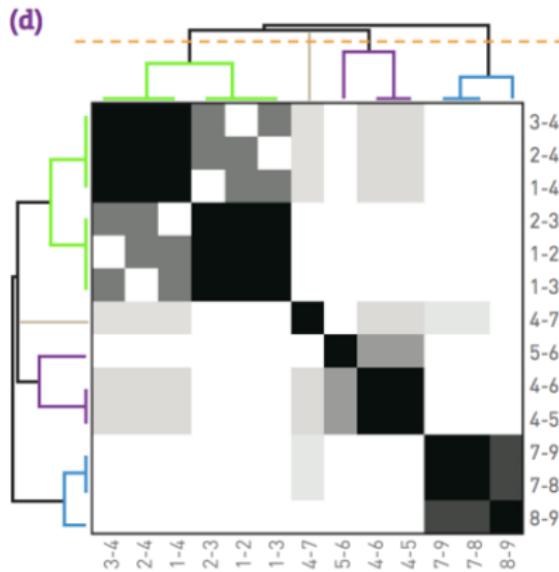
$$S(e_{ik}, e_{jk}) = \frac{|n(i) \cap n(j)|}{|n(i) \cup n(j)|}$$

dove $n(i)$ è il vicinato del nodo i incluso esso stesso.

S misura il numero relativo di vicini in comune.

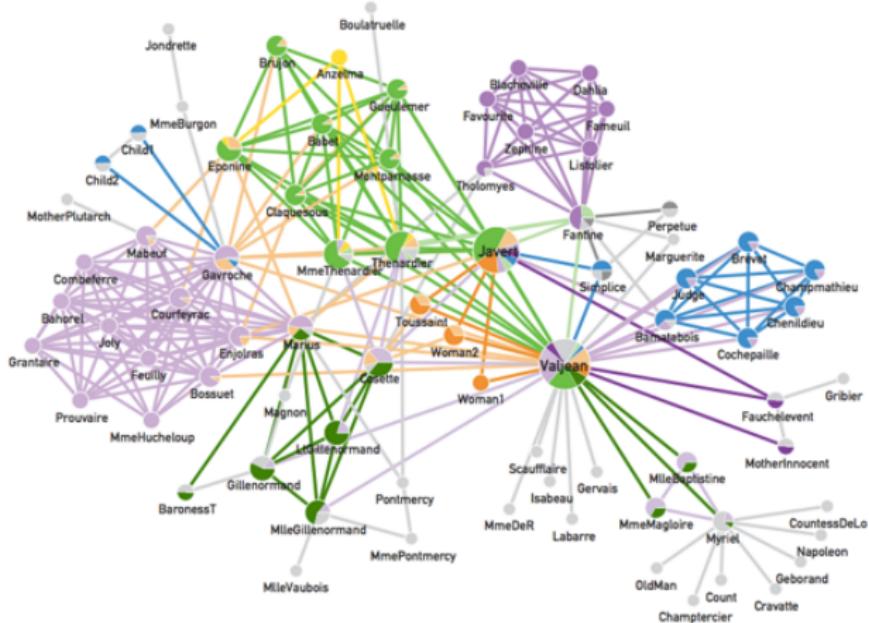


Passo 2: clustering gerarchico agglomerativo single linkage



Link clustering

La rete dei personaggi del romanzo di Victor Hugo del 1862 Les Misérables. Due personaggi sono collegati se interagiscono direttamente tra loro nella storia. I colori dei collegamenti indicano i cluster, nodi grigi corrispondenti ai cluster a collegamento singolo. Ogni nodo è rappresentato come un grafico a torta, che illustra la sua appartenenza a più comunità. Non sorprende che il personaggio principale, Jean Valjean, abbia i membri della comunità più diversificata



Approccio della teoria dell'informazione: se due partizioni sono simili, una ha bisogno di pochissime informazioni per dedurre una partizione data l'altra. Possiamo utilizzare l'**informazione mutua**:

$$I(\{C_1\}, \{C_2\}) = \sum_{C_1} \sum_{C_2} p(C_1, C_2) \log \frac{p(C_1, C_2)}{p(C_1)p(C_2)}$$

Probabilità congiunta che un nodo scelto casualmente appartenga alla comunità C_1 nella prima partizione e C_2 nella seconda:

$$p(C_1, C_2) = \frac{\text{numero di nodi che sono sia in } C_1 \text{ che in } C_2}{\text{tutte le possibili coppie tra } C_1 \text{ e } C_2} = \frac{N_{C_1 C_2}}{\sum_{C_1, C_2} N_{C_1 C_2}}$$

Probabilità che un nodo scelto casualmente appartenga alla comunità C_1

$$p(C_1) = \frac{\text{numero di nodi appartenenti a } C_1}{\text{somma su tutte le partizioni}} = \frac{N_{C_1}}{\sum_C N_C}$$

L'informazione reciproca non è ideale come misura di somiglianza così com'è: infatti, data una partizione C, tutte le partizioni derivate da C partizionando ulteriormente i suoi cluster avrebbero tutte la stessa informazione reciproca con C, anche se potrebbero essere molto diverse l'una dall'altra.

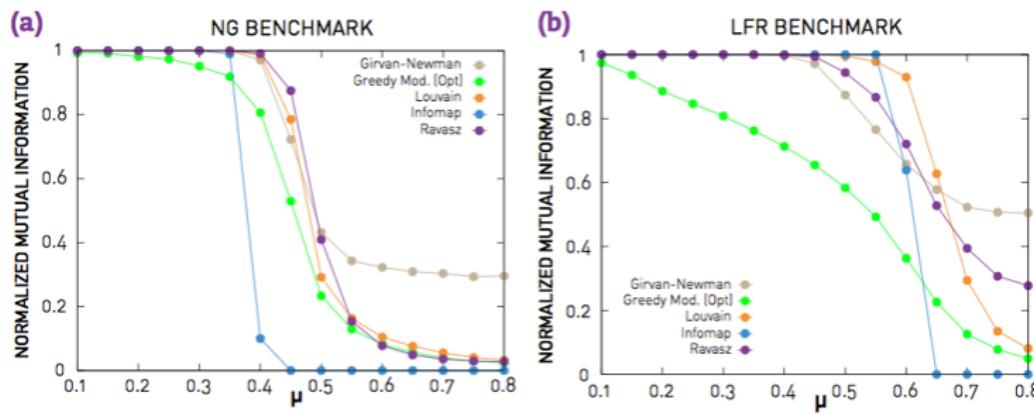
In questo caso l'informazione reciproca sarebbe semplicemente uguale all'entropia $H(X)$, perché l'entropia condizionale sarebbe sistematicamente zero. Inoltre la misura non è normalizzata.

$$I_n(\{c_1\}, \{C_2\}) = \frac{2I(\{c_1\}, \{C_2\})}{H(\{c_1\}) + H(\{C_2\})}$$

Per:

- $I_n = 1$ le comunità sono identiche
- $I_n = 0$ le comunità sono indipendenti

- N nodi nelle comunità N_C
- Ad ogni nodo viene assegnato un grado k tale che $P_I \sim k^\lambda$
- Ogni nodo i è assegnato a una comunità, e riceve un grado interno $(1 - \mu)k_i$ e un grado esterno μk_i .
- Connotti i nodi in modo casuale, secondo i vincoli di cui sopra, fino a quando non ci sono più stub liberi



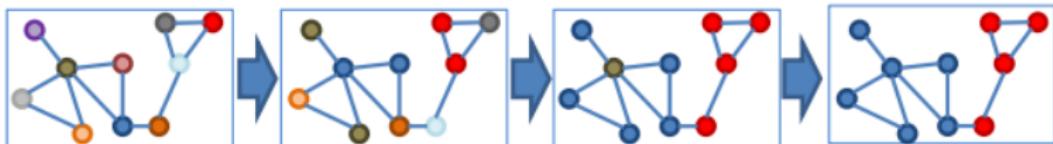
È un algoritmo veloce per la ricerca di comunità in un grafo.

- Rileva le comunità utilizzando solo la struttura della rete come guida e
- non richiede una funzione obiettivo predefinita o
- informazioni preliminari sulla struttura della rete o sulle comunità

Label Propagation Algorithm (LPA - 2007)

L'algoritmo procede in round

- Inizialmente, a ciascun vertice della rete viene assegnata un'etichetta univoca (è la propria comunità iniziale)
- Le etichette si propagano attraverso la rete.
- Ad ogni round della propagazione, ciascun nodo aggiorna la sua etichetta a quella a cui appartiene il numero massimo di vicini
- LPA raggiunge la convergenza quando ogni nodo ha l'etichetta maggioritaria tra suoi vicini.
- LPA termina se viene raggiunta la convergenza o il numero massimo di iterazioni definito dall'utente.



Label Propagation Algorithm

Algorithm 1: Label Propagation (Synchronous)

```
1 Inizialize labels: for each  $v \in V$ ,  $l_v(0) = v$ ;  
2  $i=0$ ;  
3 while the stop criterion is not met do  
4    $i++$ ;  
5   Propagation:  
6   foreach  $v \in V$  do  
7      $l_v(i) = \operatorname{argmax}_l \sum_{u \in N(v)} [l_u(i-1) == l]$ ,  
8   end  
9 end  
10 return Final labeling:  $l_v(t)$  for each  $v \in V$ , where  $t$  is  
the last executed step.
```

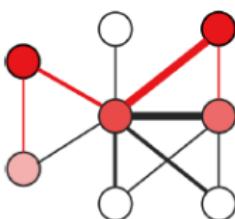
Note (valide anche per il seguito).

Nell'esempio l'etichetta di un nodo è rappresentata dal colore del nodo stesso

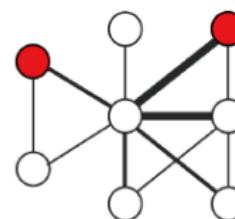
Nell'algoritmo $[x==y]$ assume valore 1 se $x=y$, valore 0 se gli argomenti sono diversi

Label Propagation

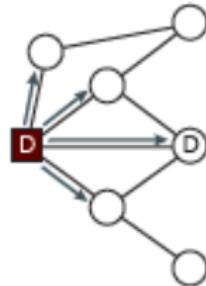
Direct Neighborhood



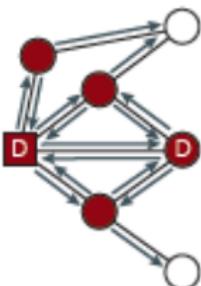
Network Diffusion



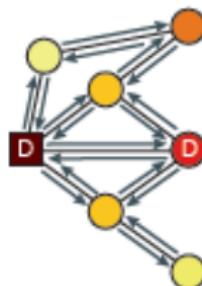
a t=0



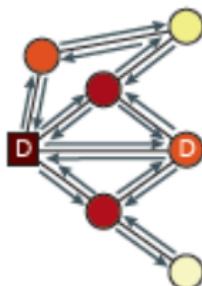
t=1



t=2



t=3



...

t= ∞

