

# Teoria

---

## Teoria

### Complex Network Analysis

---

prof. Andrea Passarella

#### Course material

---

- ❑ Slides online in the master repository
  - ❑ A few papers available on the same website
- ❑ Igraph (Python module)
  - ❑ <http://igraph.org>
  - ❑ <https://igraph.org/python/>
- ❑ Textbooks
  - ❑ A.L. Barabási, "Network Science", 2016, <http://barabasi.com/networksciencebook/>
  - ❑ M. Newman, "Networks: An Introduction",  
<https://www.oxfordscholarship.com/view/10.1093/acprof:oso/9780199206650.001.0001/acprof-9780199206650>
- ❑ Example of other courses (typically, Social Network Analysis)
  - ❑ D. Pedreschi, UniPi, Social Network Analysis  
[http://didawiki.cli.di.unipi.it/doku.php/wma/start#social\\_network\\_analysis](http://didawiki.cli.di.unipi.it/doku.php/wma/start#social_network_analysis)

#### Housekeeping

---

- ❑ Lectures + Labs (7h + 7h)
  - ❑ Hands-on lectures
  - ❑ Use immediately what we see in lectures
- ❑ Project work (7h)
  - ❑ Develop a project based on the tools seen during lectures
- ❑ Group-work
  - ❑ groups of 2-4 people
- ❑ Final exam (3h+)
  - ❑ Discussion of the project

## Exam

---

- Project discussion
  - Short presentation with Q&A
- Project work
  - Build 9 groups, i.e., groups of ~4 people
  - Use existing groups if you have them (or change them if you prefer)
- Each group is assigned a dataset describing a network
  - Do some SNA on the dataset using the igraph tools you learn in lectures
  - Focusing on robustness and attacks
- Schedule
  - 30' per group
  - 3h on 30/07
  - another 3h earlier in the same week/ in the previous week / in the following week

## Exam

---

- Project is collaborative, so you can organise work as you like in the group
- Discussion of the project is INDIVIDUAL
- During the discussion, each of you should present a part of the project
  - NO ONE can stay silent
- You can decide who presents what
  - BUT I can ask each of you to answer questions on the entire project
  - All members of the group must be familiar with the entire project
- The grade of the project is INDIVIDUAL
  - Based on the outcome of the discussion
  - Different members of the same group can get different grades

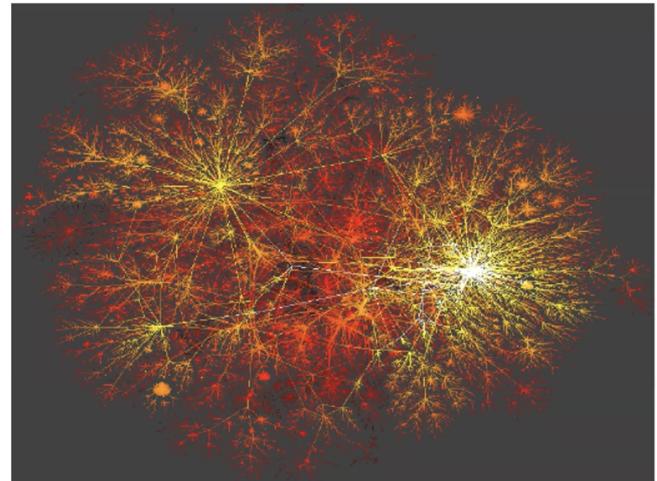
Quando parliamo di reti complessi parliamo di un modo di rappresentare sistemi fisici a larga scala che sono formati da entità generiche collegate tra di loro.

## What do we talk about ...

... when we talk about Complex Networks?

- (very) Large scale physical systems
- Composed of interconnected "entities"

The map of the Internet

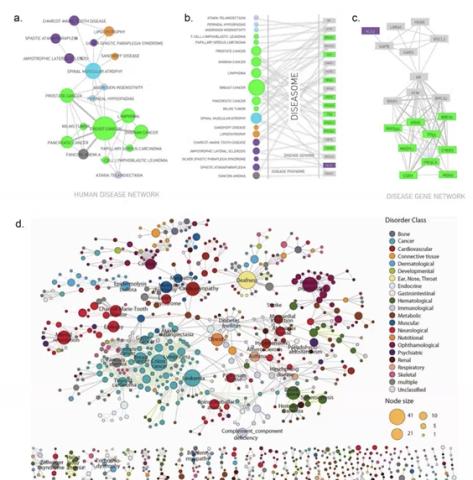


## What do we talk about ...

... when we talk about Complex Networks?

- (very) Large scale physical systems
- Composed of interconnected "entities"

Genes, Proteins, Drugs



Possiamo usare gli stessi strumenti per rappresentare ad esempio le reti elettriche europee oppure le strutture interne delle aziende oppure per sistemi naturali come il cervello, i geni, le proteine, le interazioni tra farmaci oppure sistemi relativi al dominio ambientale o animale.

Esiste una varietà di sistemi che possono essere analizzati con gli strumenti presentati in questo corso

la cui essenza della ragione sta nella seguente frase di Barabasi:

### Why do we care?

- 1 "Notwithstanding the amazing differences in form, size, nature, age, and scope of real networks, most networks are driven by common organizing principles."
- 2
- 3 Once we disregard the nature of the components and the precise nature of the interactions between them, the obtained networks are more similar than different from each other".
- 4

A. L. Barabasi, "Network Science Book"

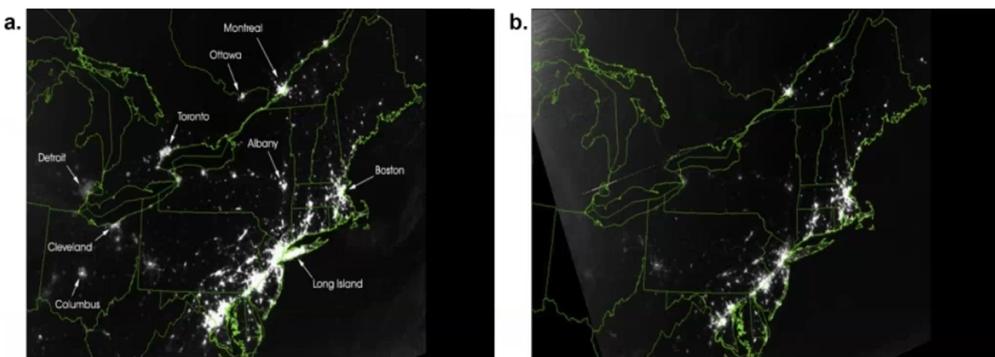
Tutti i sistemi sono differenti tra di loro ma nonostante le differenze, molti sistemi che si basano sulle reti, hanno proprietà determinate in gran parte dal fatto che presentano determinati pattern di connettività. Se riusciamo ad astrarre la specificità della semantica del sistema studiato e ci focalizziamo solo sui pattern di interconnessione, tutti i sistemi possono essere rappresentati tramite reti molto simili tra di loro.

Esistono famosi esempi ad esempio sui sistemi elettrici a larga scala in cui è stato dimostrato che è possibile capire qual'è l'effetto di rottura di una centrale di alimentazione di New York su tutta la zona di Manhattan e dintorni:

### Why do we care?

#### Vulnerabilities

- Power grid outages



Si possono quindi usare per studiare aspetti di fragilità e robustezza, per capire l'effetto sulla rete complessiva a seguito di un attacco di uno o più specifici nodi:

## Why do we care?

---

### Fragility vs robustness

- Internet robustness



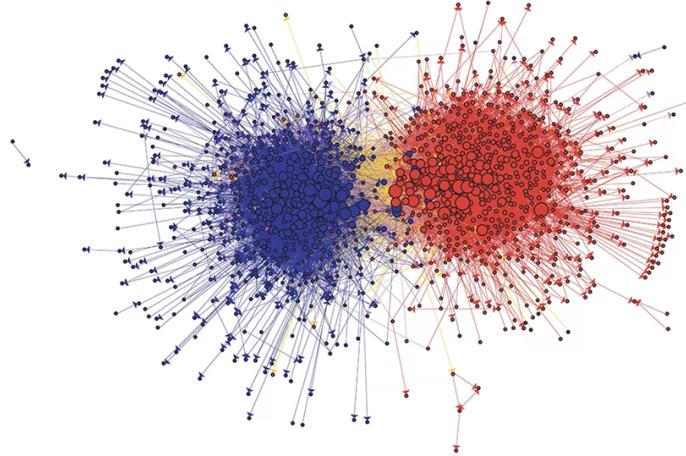
Con questi strumenti si riescono ad analizzare le dinamiche delle opinioni e di polarizzazione sui social networks:

## Why do we care?

---

### Opinions

- Polarisation



Tutti questi sistemi possono essere rappresentati tramite i **grafi**:

## How does it work?

---

- ❑ Key: **Graphs**
  - ❑ All the above systems can be represented with one unique abstraction
- ❑ A set of standard **mathematical tools** to analyse graphs
  - ❑ Very simple indices to represent global properties of the underlying system
  - ❑ Compact representation of structural properties of possibly huge graphs
- ❑ These properties **determine**, by and large, key **phenomena** of the analysed system
  - ❑ Resilience, epidemics, communities, ...
  - ❑ **Evolution** over time

sul quale grafo si possono usare dei **tool matematici**, degli indici, dei numeri che si ricavano dal grafo che ci permettono di rappresentare proprietà globali del grafo stesso come ad esempio la resilienza, la robustezza, l'evoluzione nel tempo:

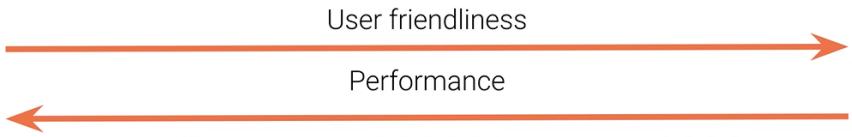
## Software Tools

---

- ❑ Dedicated software (Gephi, Pajek, ...)
  - ❑ Good solutions for quick analyses of small networks
  - ❑ Simple, intuitive, and with great visualization
  - ❑ Suited only for small-scale networks
- ❑ Big data oriented tools (Hadoop, Giraph, ...)
  - ❑ Top performances
  - ❑ Best solution for very large networks and production environments
  - ❑ Very specific and with limited functions
- ❑ Specialized libraries for programming languages (graph-tool for Python, igraph for Python and R, SNAP)
  - ❑ Wide range of options
  - ❑ Good performances
  - ❑ Large number of functions

Esistono diversi tool software e librerie per i linguaggi di programmazione. Nell'immagine c'è un esempio di diversi tool e di come questi rispondono in base al tempo necessario per calcolare gli indici:

Is there a single “right” tool?



Algorithm	graph-tool (4 cores)	graph-tool (1 core)	igraph	NetworkX
Single-source shortest path	0.004 s	0.004 s	0.012 s	0.152 s
PageRank	0.029 s	0.045 s	0.093 s	3.949 s
K-core	0.014 s	0.014 s	0.022 s	0.714 s
Minimum spanning tree	0.040 s	0.031 s	0.044 s	2.045 s
Betweenness	244.3 s (~4.1 mins)	601.2 s (~10 mins) + 353.9 s (vertex) (~ 21.6 mins)	946.8 s (edge) 22650.4 s (vertex) (~15.4 hours)	32676.4 s (edge)

gli strumenti come graph-tool nativamente implementati in C/C++ sono prestazionalmente più efficienti rispetto agli altri tool.

### Topics covered

- ❑ CNA ([math](#)) tools
  - ❑ Set of indices and algorithms to analyse structural properties of large networks
- ❑ The [igraph](#) python library for Complex Network Analysis
  - ❑ <http://igraph.org/python>
  - ❑ Core written in C++
  - ❑ Good performances and easy to use
  - ❑ The most complete CNA package, available for several languages
  - ❑ Python, R, C++

## Network & Graphs

# Complex Network Analysis

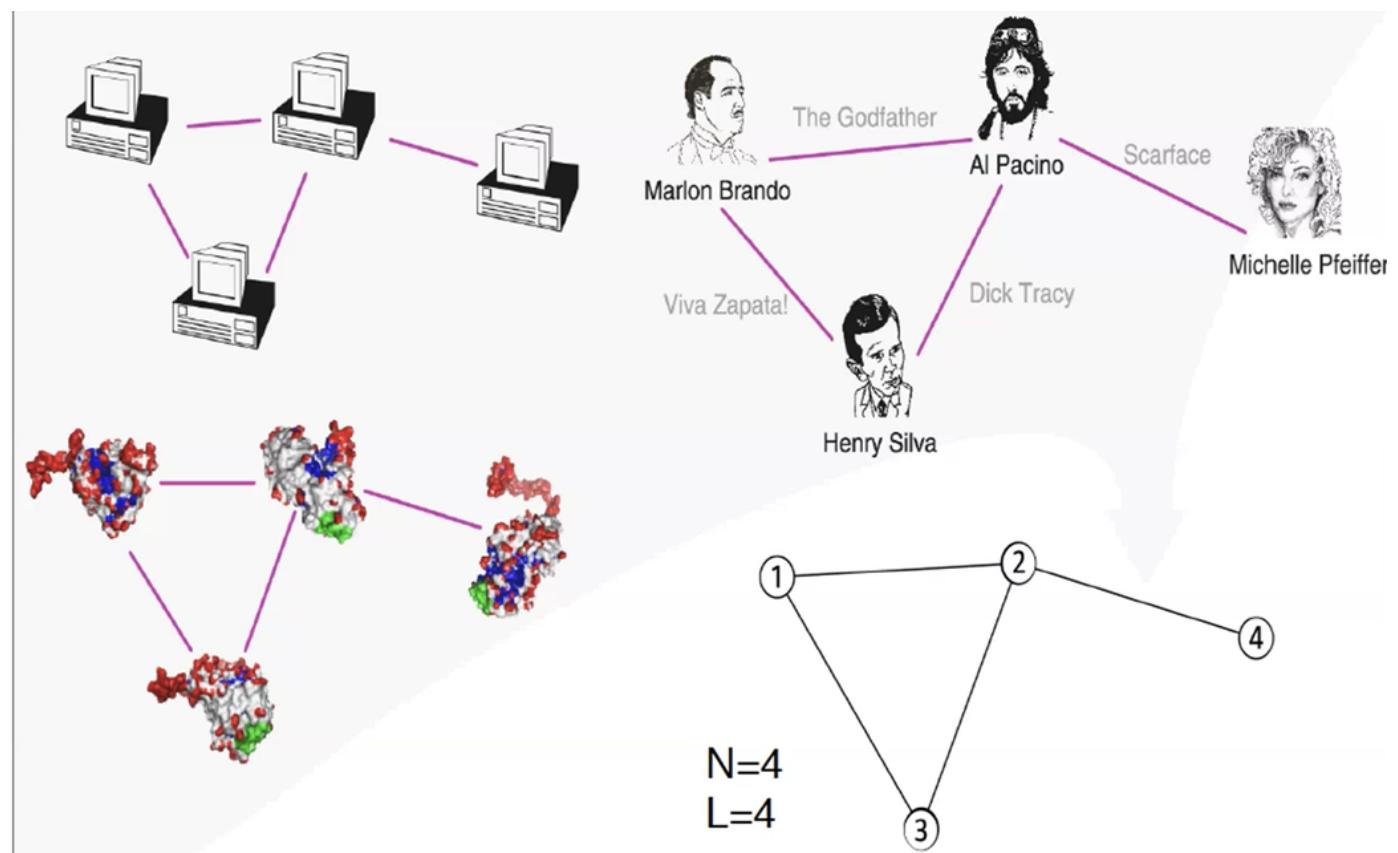
## Networks & Graphs

Andrea Passarella

a.passarella@iit.cnr.it



I  
le reti sono la forma che assumono molti sistemi reali mentre i grafi sono un modo di rappresentarli. I tool analitici sono stati visualizzati sui grafi. I 2 termini vengono poi usati come sinonimi anche se sono due concetti diversi.



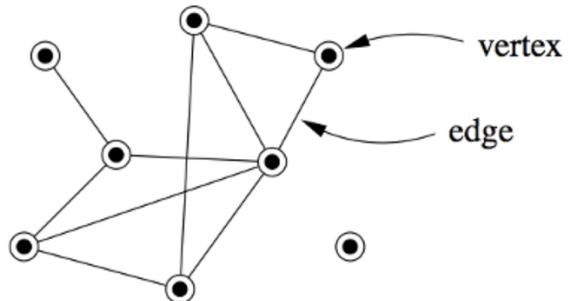
Network Science: Graph Theory January 24, 2011

Queste 3 reti da un punto di vista della connessione tra i nodi sono esattamente equivalenti anche se i sistemi che formano le reti dai quali sono stati dedotti i grafi sono completamente differenti.

## Graphs

---

- ❑ Key elements of a graph
  - ❑ Set of **vertices** (nodes)
  - ❑ Set of **edges** connecting them
  - ❑ Possibly, **weights** associated to edges



in alcuni casi vengono aggiunti degli attributi sul vertex o sull'edge. In una rete sociale, se rappresentiamo una rete sociale, possiamo rappresentare i link con un peso per esprimere la forza dell'amicizia tra due persone.

## Interesting Graphs

---

- ❑ Interesting Graphs are typically **VERY large** and **complex**
  - ❑ Internet: 200K nodes, 600K links
  - ❑ Facebook: 1.4 B active users per month, 680K mobile users
  - ❑ Twitter: 115M active users per month, about 40% are (also) mobile
  - ❑ Complex patterns of interconnections
- ❑ This poses dramatic **challenges** in understanding and evaluating their structure



## Complex Network Analysis

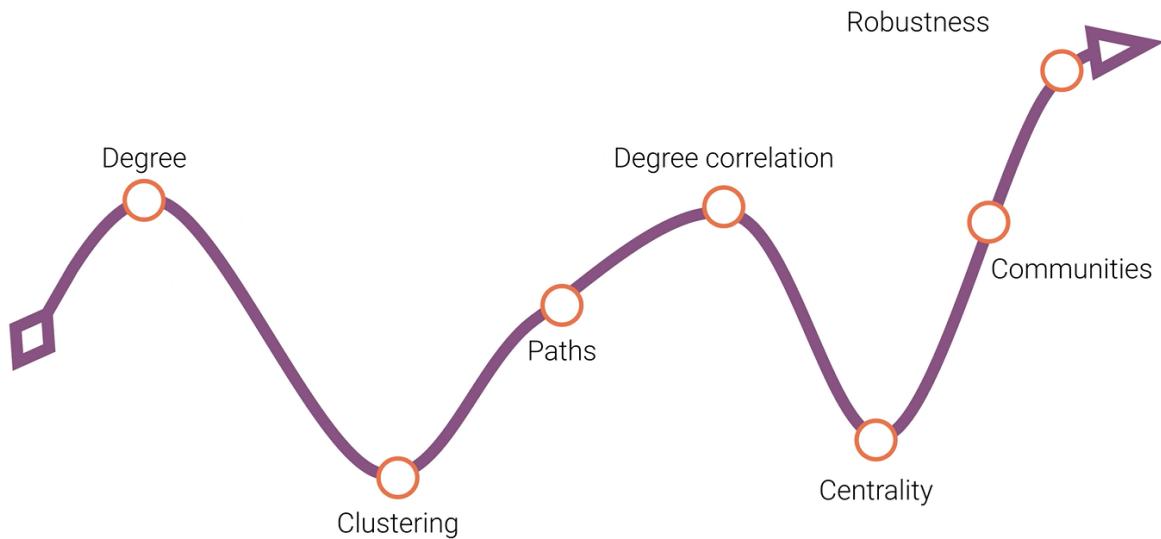
---

- ❑ Defines a set of methodologies and indices to analyse large-scale graphs
- ❑ Indices provide indications about particular features of the graph
- ❑ Depending on the value or type of certain indices, particular properties of the network can be directly derived
- ❑ Indices can be typically computed with efficient algorithms included in most data analysis tools
  - ❑ E.g., igraph and R, <http://igraph.org/>

I vari indici da analizzare sono i seguenti:

### Roadmap

---



## Degree Analysis

---

Le reti sono la forma che assumono molti sistemi reali mentre il grafo è l'astrazione di una rete. I tool analitici sono stati sviluppati sulla base dei grafi e non delle reti.

## Networks and Graphs

- ❑ Networks are the typical form of many real systems

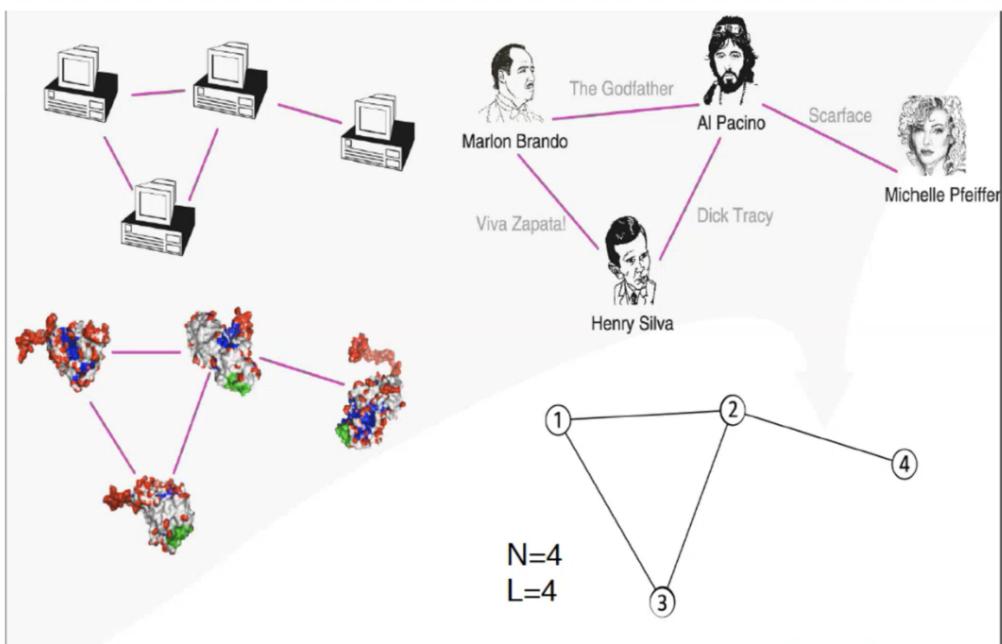
- ❑ WWW
- ❑ Internet
- ❑ social networks
- ❑ Metabolic networks
- ❑ ...

- ❑ Graphs are a common way of representing them

- ❑ general abstraction of networks
- ❑ for which analytical tools have been developed

Le differenti reti che possiamo avere nella realtà (la rete di attori, di computer o di proteine) possono essere rappresentate con il grafo in basso a destra:

## Networks and Graphs



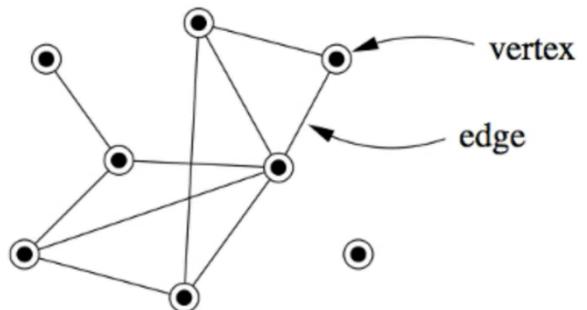
Network Science: Graph Theory January 24, 2011

Una volta rappresentata una rete tramite un grafo, abbiamo astratto e riusciamo a usare gli strumenti matematici facilmente ma la realtà ritorna a presentarsi quando dobbiamo fare alcune considerazioni. Ad esempio se vogliamo attaccare una rete di calcolatori come quella mostrata in figura, potremmo pensare di attaccare il nodo 2. Se capiamo che il nodo critico è il nodo 2 ma la nostra rete è una rete di attori, dobbiamo andare a sparare in testa Al Pacino che è diverso da isolare un computer rimuovendo le sue interfacce di rete.

? Dunque un grafo cosa è?

## Graphs

- ❑ Key elements of a graph
  - ❑ Set of **vertices** (nodes)
  - ❑ Set of **edges** connecting them
  - ❑ Possibly, **weights** associated to edges



E' un insieme di vertici o nodi e un insieme di edge o link che li collegano. In alcuni casi si aggiungono degli attributi sia agli edge che ai vertici. Normalmente si trovano dei pesi sui link per indicare un peso differente per edge differenti.

## Interesting Graphs

- ❑ Interesting Graphs are typically **VERY large** and **complex**
  - ❑ Internet: 200K nodes, 600K links
  - ❑ Facebook: 1.4 B active users per month, 680K mobile users
  - ❑ Twitter: 115M active users per month, about 40% are (also) mobile
  - ❑ Complex patterns of interconnections
- ❑ This poses dramatic **challenges** in understanding and evaluating their structure



I grafi che rappresentano sistemi reali sono molto complessi.

La complex network analysis ci permette di astrarre la realtà con un grafo, di applicare degli indici che permettono di gestire la scala complessa del sistema che vogliamo analizzare e infine usiamo il toolbox

per calcolare i vari indici.

## Complex Network Analysis

- ❑ Defines a set of **methodologies** and **indices** to analyse large-scale graphs
- ❑ Indices provide indications about particular **features** of the graph
- ❑ Depending on the value or type of certain indices, particular **properties** of the network can be directly derived
- ❑ Indices can be typically computed with **efficient algorithms** included in most data analysis tools
  - ❑ E.g., igraph and R, <http://igraph.org/>

Partiamo dal **grado dei nodi** all'interno della rete.



### Complex Network Analysis Degree Analysis

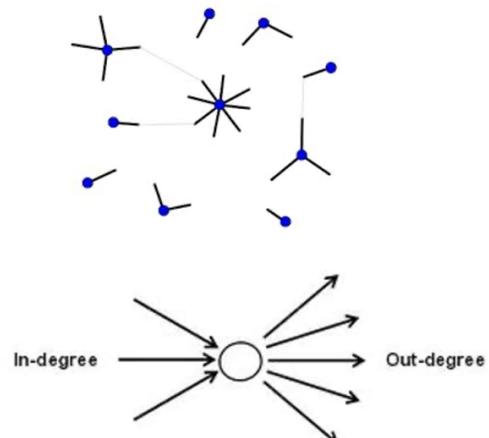
Andrea Passarella  
[a.passarella@iit.cnr.it](mailto:a.passarella@iit.cnr.it)



Ogni indice ci da un punto di vista sul sistema. Si parla di sistemi in cui non esiste un indice che racconta la storia completa del sistema, questi che vedremo sono solo modi di guardare una rete da punti di vista diversi. Dunque una rete si può analizzare dal punto di vista del grado, del clustering, si può analizzare dal punto di vista dei path, etc...

## Degree of a node

- ❑ Number of edges in/out the node
- ❑ In-degree
  - ❑ Number of edges getting into the node
- ❑ Out-degree
  - ❑ Number of edges going out the node
- ❑ In general
  - ❑ In- and out-degree can be way different
  - ❑ E.g., Twitter
    - ❑ #follower = in-degree
    - ❑ #following = out-degree
- ❑ When not specified
  - ❑ Assumption that edges are bidirectional
  - ❑ In-degree = out-degree



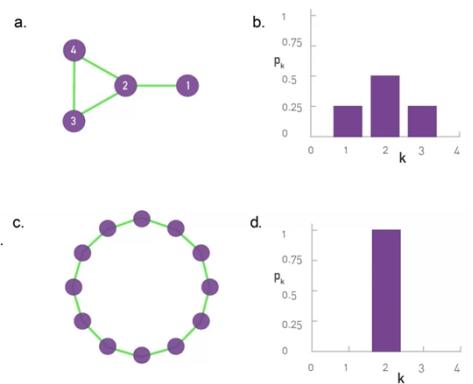
**Il grado di una rete è il numero di link o edge che entrano o escono da un nodo.** Un'altra informazione che si mette nella rappresentazione del grado è la direzione ovvero gli edge di un nodo possono essere entranti o uscenti (in-degree vs out-degree). Questa direzionalità degli archi da un punto di vista macroscopico potrebbe non essere poi così importante.

Per tante proprietà non è importante capire quanti in-degree e out-degree ci sono, ma il numero totale. Mentre per alcuni indici è importante conoscere la direzionalità.

L'analisi del grado di una rete si fa tramite l'analisi della **distribuzione**:

## Degree Distribution

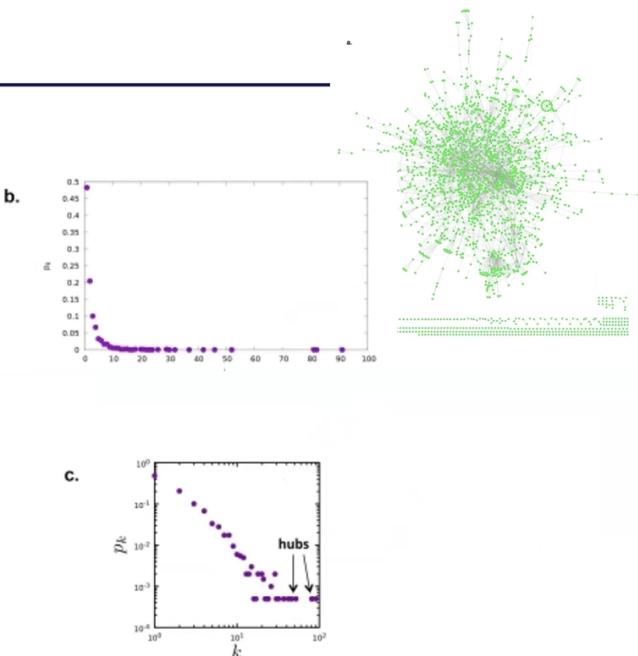
- ❑ Fraction of nodes with a certain degree
  - ❑ Precisely, this is the empirical density
- ❑ Fraction of nodes ~ probability of finding a node with that degree
  - ❑ a. Probability of nodes having degree 2:  $p_2 = 0.5$



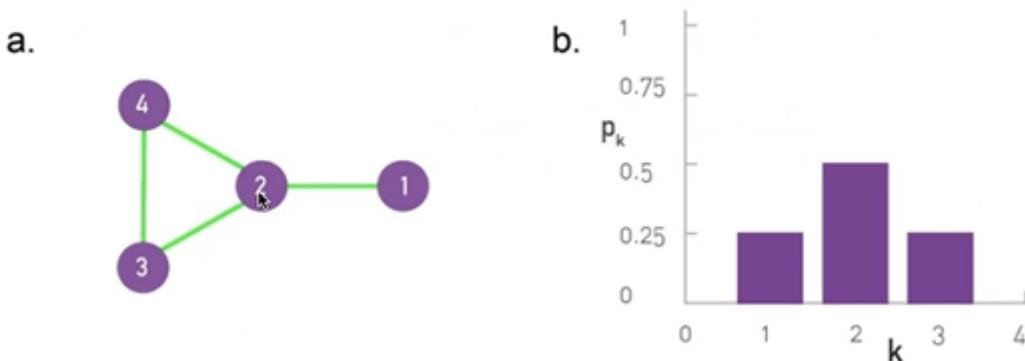
Supponiamo di avere una rete come nella figura di esempio a e c e ci calcoliamo la frazione o percentuale di nodi con un certo grado. Nella rete c essendo circolare ho il 100% di nodi con grado 2. Questa appena calcolata è detta **DENSITA' EMPIRICA DEL GRADO DELLA RETE**, che corrisponde a calcolare il numero di nodi con un certo grado nella mia rete e dividere questo numero per il numero totale di nodi ottenendo la frazione di nodi che hanno quel particolare grado.

## Hubs

- ❑ Most interesting part of the distribution is the **tail**
  - ❑ Distribution for **large values** of degree
  - ❑ This is where we find "**hubs**"
- ❑ Hubs = nodes with "**very high**" degree
- ❑ If they exist
  - ❑ They are **only a few**
  - ❑ They have a degree **much larger** than the "normal" nodes



La parte più interessante di questi grafici empirici è quella che si chiama la **coda**, ovvero i valori che troviamo in queste curve per valori alti del grado. Nel caso **b** è stato fatto esattamente lo stesso esercizio precedente, ma la parte interessante di questo grafico **b** sono i valori sulla destra per valori alti nel grado. Ci interessa capire la probabilità di avere nodi ad alto grado perché questi sono detti **HUB**, ovvero **nodi con il grado più alto in una rete (in una rete sociale possono essere un politico, un influencer, etc...)**. La coda della degree distribution ci dice quali hub ha la nostra rete e ci dice anche se esistono. Per definizione gli hub sono nodi a grado massimo, ma volendoci basare solo su questa definizione, anche sulla rete **a** dell'immagine precedente possiamo identificare un hub:

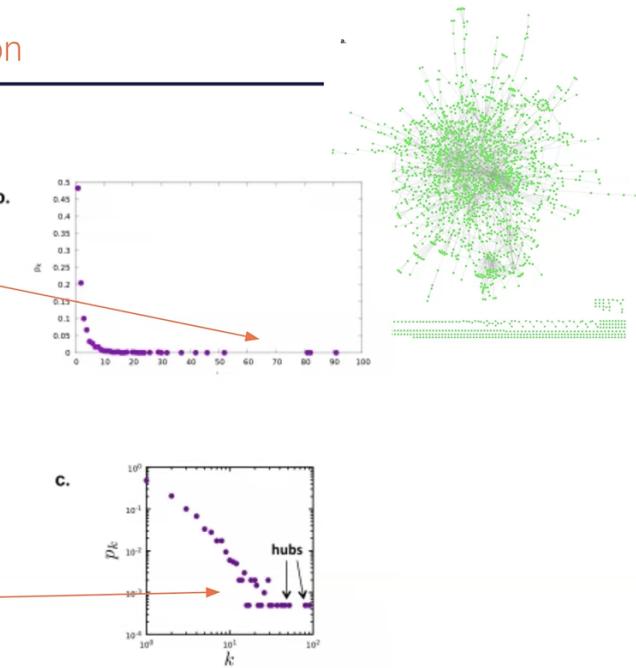


l'hub è 2 in quanto nodo a grado massimo ma non si parla di hub in quanto il grado del nodo 2 è **molto vicino** al grado degli altri nodi.

❓ Come viene studiata questa parte di distribuzione?

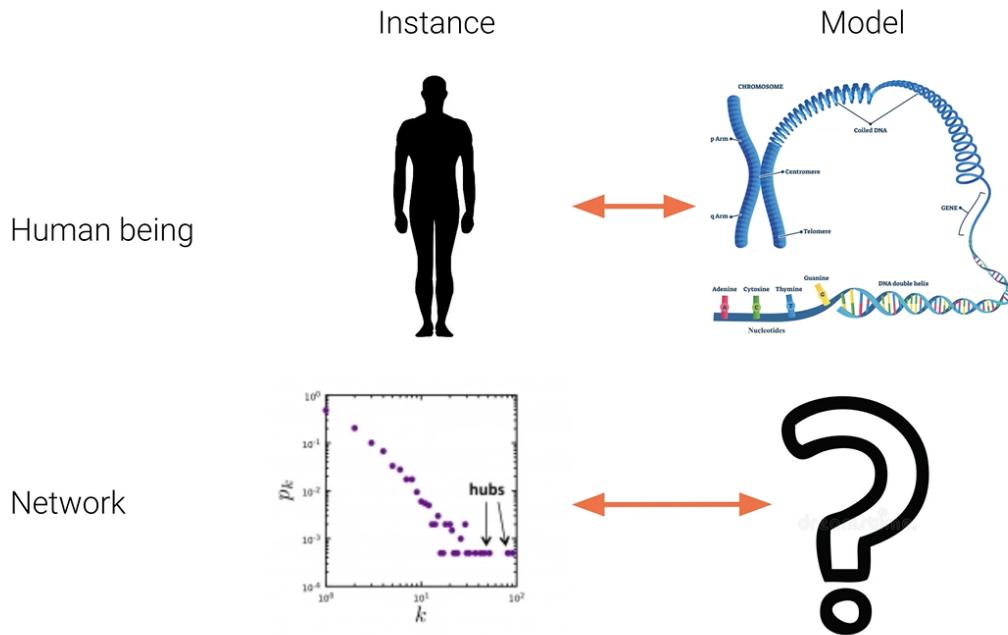
## Visualising hubs via the degree distribution

- ❑ “Normal” ([linear](#)) way of visualising a distribution is not helpful
  - ❑ Curve is typically very flat and differences are hard to notice
- ❑ “[Log-log](#)” scale is used to highlight the shape of the curve [in the tail](#)
  - ❑ Plot  $\log(k)$  vs  $\log(P_k)$



Il modo normale è fare esattamente quello già spiegato, ovvero produrre **gli istogrammi con le probabilità**. Il plot [b](#) però ha sulla destra una riduzione delle differenze nella curva, dunque la curva si appiattisce e le differenze non si notano in modo deciso, questo perchè gli hub per essere diversi da tutti gli altri altri devono essere **molto pochi**. Gli hub di una rete devono avere un grado molto maggiore ed ipso facto devono essere pochi rispetto al numero complessivo di nodi della rete. Dunque anzichè plottare la probabilità di avere nodi a grado  $K$  con una scala lineare si plotta una **scala logaritmica**. Il grafico [c](#) rappresenta esattamente gli stessi dati del grafico [b](#) ma visualizzati su una scala detta "log x log" rappresentando la relazione tra il logaritmo del grado e il logaritmo della probabilità di ottenere quel grado. La compressione della scala logaritmica espande alcune zone dell'asse e comprime altre zone per permettere di vedere chiaramente i dati in modo più separato.

## Are the empirical data about the distribution enough?



Questa metodologia vista fin ora consiste nel dire: ho la mia rete, mi calcolo quanti nodi hanno un certo grado, divido per il numero di nodi e ottengo la densità di probabilità. Questa operazione corrisponde a fare una fotografia della rete in un certo istante della sua vita. E' un po come fare la fotografia di una persona in un momento particolare della sua vita. Per alcune situazioni è sufficiente ma per altre no. Per avere maggiori informazioni su come un essere umano evolverà nel tempo per capire se svilupperà eventuali malattie o meno bisogna guardare il genoma della persona. Quindi calcolare la densità empirica in un certo momento di una rete equivale a fargli una fotografia in quell'istante temporale, ma in alcune situazioni voglio cogliere delle proprietà strutturali della rete più generiche, ovvero avere uno strumento che non mi descrive solo lo stato della rete in quel momento ma mi da delle proprietà più intrinseche della rete che non deduco immediatamente dalla rete.

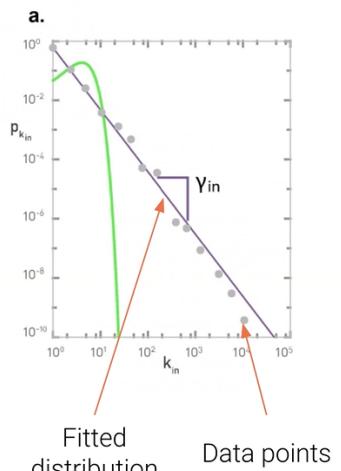
💡 L'equivalente del genoma per quanto riguarda l'analisi del grado sono la densità non empirica e poi la distribuzione.

Per calcolare questi indici si usa uno strumento detto **FITTING**

## Fitting distributions

### □ Fitting

- find the best mathematical formula that approximates the empirical data
- The network may change over time
  - But the fitted distribution (if done properly) will not
- Analyse the fitted distribution (and not the data)

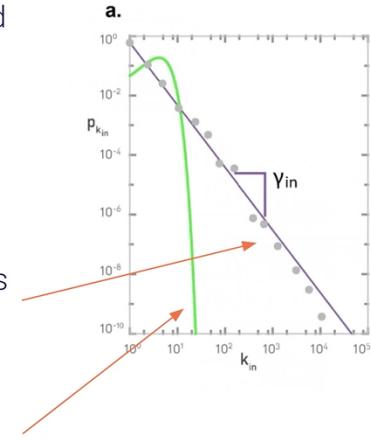


Il fitting vuol dire che ho dei dati empirici e cerco di metterci sopra una legge matematica che me li approssimi nel miglior modo possibile, una sorta di "interpolazione evoluta". Nel plot a destra ho i miei dati che empiricamente ho calcolato dalla rete e voglio trovare una funzione matematica che me li approssimi correttamente. **Il fitting non varia una volta che il sistema arriva a regime.**

Questa legge matematica che viene ottenuta dal fitting dei dati permette di passare da una fotografia al genoma della rete. Questo permette di capire a seconda della legge matematica che fitta i miei dati di capire alcune cose:

## Heavy vs light tails

- For all networks,  $p_k$  decreases with  $k$ 
  - The probability of having large degrees gets smaller and smaller as  $k$  increases
- However, it is important to analyse "how fast" the probability decreases
- **Heavy tails**
  - The tail decreases slow  $\Rightarrow$  probability of large degrees is relatively high
- **Light tails**
  - The tail decreases fast  $\Rightarrow$  probability of large degrees is relatively low



Prendendo come esempio la distribuzione di pallini grigi mostrati in figura, supponendo che potremo scegliere come legge di fitting solo tra quella viola e quella verde, troveremo che è quella viola che fitta meglio i miei dati.

La distribuzione viola e verde sono le 2 distribuzioni che meglio fittano i miei dati all'interno di due famiglie distinte: la curva viola è il fitting migliore dei dati nella famiglia di distribuzioni a **coda pesante** mentre quella verde è il fitting migliore dei dati nella famiglia di distribuzioni a **coda leggera**. Le

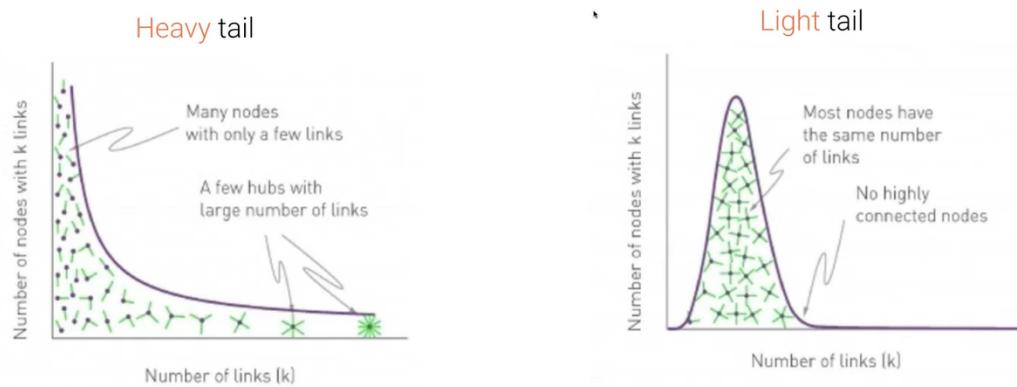
distribuzioni a coda pesante hanno una probabilità di avere nodi a grado alto che decresce man mano che descresce il grado, ma lo fa in modo lento. Dunque le distribuzioni a coda pesante hanno un decadimento molto lento ovvero c'è una probabilità relativamente alta di avere nodi a grado alto. Le distribuzioni a coda leggera (curva verde) hanno un decadimento più veloce.

**⚠ Se trovo dunque che i miei dati vengono fittati meglio da una distribuzione a coda pesante, allora la mia rete o gli hub ce li ha già e li osservo o se non li ha è molto probabile che li svilupperà in futuro in quanto il suo genoma è a coda pesante.**

*Se il fitting è migliore per una curva della famiglia a coda leggera, allora la mia rete non ha hub, perché alcuni nodi hanno grado maggiore ma non tanto maggiore rispetto agli altri e dunque la rete non è predisposta a sviluppare hub in futuro.*

## Heavy vs light tails

---



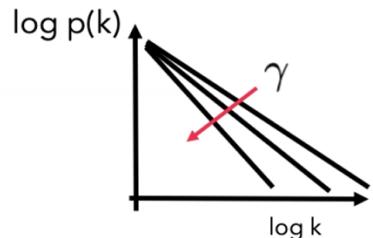
Se le distribuzioni sono a coda leggera, la probabilità di avere nodi a grado alto decresce velocemente dunque non c'è spazio per avere una probabilità sufficiente per avere nodi con tanti link (hub).

## Power Law Distributions

## Power law distributions

---

- Paradigmatic distribution with **heavy** tail
  - Analytically,  $p(k) = Ck^{-\gamma}$ ,  $\gamma > 1$
- Characteristic feature: a straight line in a log-log plot
  - $\log p(k) = -\gamma \log k + \log C$
  - Where the exponent  $\gamma$  is the angular coefficient of the line
  - The **lower**  $\gamma$ , the **heavier** the tail



La Power Law è definita come la probabilità di avere nodi di grado  $k$  è uguale a una costante  $C$  moltiplicato per  $k$  elevato a una potenza negativa  $-\gamma$ . Ha una proprietà visiva comoda in cui pottandola in un plot log x log otteniamo una retta che decade con un coefficiente angolare pari a  $\gamma$ , il ruolo di questo  $\gamma$  fa sì che più alto è  $\gamma$ , più la curva si schiaccia e più la coda è leggera. Se troviamo che i nostri dati sono fittati bene da una power law,  $\gamma$  ci darà il peso della distribuzione. Questo unico indice  $\gamma$  ci racconta molte cose sulla rete perché ci fa capire quale è la predisposizione della rete a sviluppare degli hub.

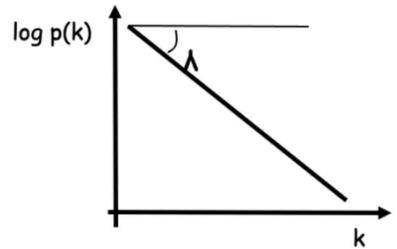
## Poisson/Exponential Distribution

Se invece faccio il fitting e la mia distribuzione è a coda leggera avrò una distribuzione esponenziale o di Poisson:

### Poisson/Exponential Distributions

---

- Paradigmatic distribution with **light** tail
  - Analytically,  $p(k) = \lambda e^{-\lambda k}$
- Characteristic feature: a straight line in a lin-log plot
  - $\log p(k) = -\lambda k + \log \lambda$
  - Where the exponent  $\lambda$  is the angular coefficient of the line



ovvero la probabilità di  $k$  è una funzione esponenziale di  $k$  con un parametro  $\lambda$ .

Questa volta si tratta nuovamente di una retta ma in un plot lin x log. Infatti calcolando il logaritmo ad

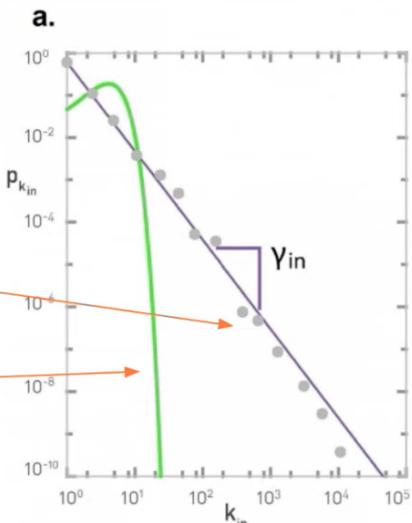
entrambi i membri della equazione otteniamo l'equazione  $\log p(k) = -\lambda k + \log \lambda$ . Il  $\log \lambda$  non ci interessa mentre l'altra parte è una linea retta decrescente con coefficiente  $-\lambda$ .

**💡 Il modo per capire con una rule of thumb se la rete verrà fittata da una distribuzione a coda leggera o pesante è prendere i dati empirici e metterli in una scala log x log oppure lin x log e vedere quale scala fitta meglio la distribuzione dei dati.**

## Heavy vs light tails

---

- Log-log scale
- Power law (heavy tail)
  - Slow decay, straight line
- Exponential/Poisson law (light tail)
  - Fast decay



Nelle reti reali esiste un unico caso famoso di reti a coda leggera: le reti elettriche.

Questo perché collegare reti elettriche non è un compito semplice dunque è naturale che su reti di questo tipo la dimensione degli hub non sia esagerata in quanto è molto costoso collegare tante sottocentrali ad una centrale unica.

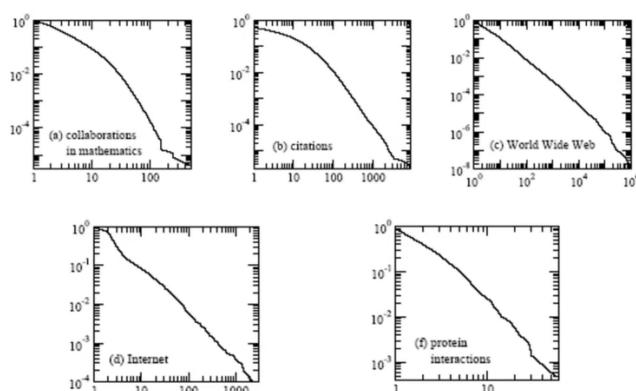
Su altre reti è normale invece trovare code pesanti.

---

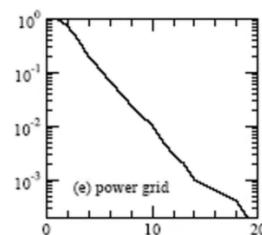
## Real Networks

---

### Heavy-tailed networks

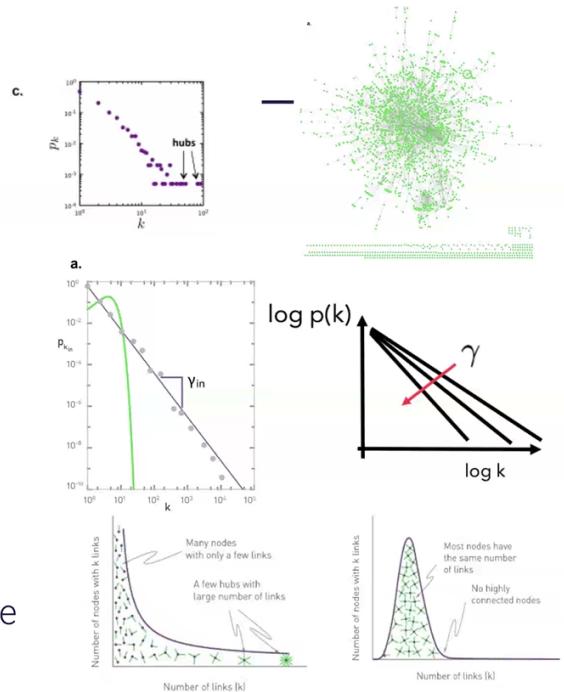


### Light-tailed networks



## Hubs and degree distribution: summary

- ❑ Hubs are important
- ❑ The degree distribution tells us
  - if hubs exist
  - how big they can be
- ❑ Heavy tail (power law)  $\Rightarrow$  large hubs
  - The lower  $\gamma$ , the bigger the hubs
- ❑ Light tail (exponential/poisson)  $\Rightarrow$  no hubs
- ❑ Log-log plot to visually tell which is the case



Gli hub sono nodi a grado alto e con un grado medio molto maggiore rispetto agli altri. Si usa la degree distribution per trovare una legge di distribuzione power law (rete predisposta a presentare degli hub) per code pesanti oppure una legge di distribuzione esponenziale (rete predisposta a non presentare degli hub) per code leggere.

19/05/2023

Un altro modo di vedere la stessa cosa è capire quale è l'impatto della distribuzione del grado sulla variabilità dei gradi stessi.

Si può dimostrare che le formule che descrivono il grado massimo di una rete rispettivamente power law e di poisson sono quelle mostrate di seguito:

## Degree distribution and degree variability

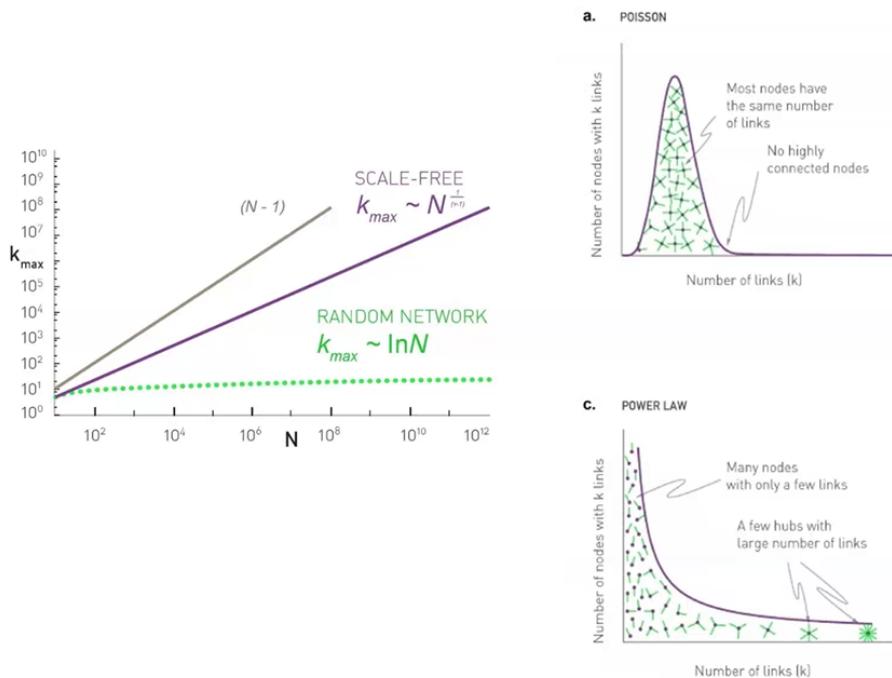
- ❑ The degree distribution determines (among others) how much **variability** one observes in the degree among nodes
  - typically captured by the difference between the minimum ( $k_{\min}$ ) and maximum ( $k_{\max}$ ) degree present in the network
- ❑ For typical **non-heavy tailed** distributions  $k_{\max} = k_{\min} + \frac{\ln N}{\lambda}$ 
  - $\ln(N)$  increases very slowly with  $N$
  - $k_{\min}$  and  $k_{\max}$  are not very different
  - degrees are quite concentrated around the average value,  $\langle k \rangle$
- ❑ For typical **heavy-tailed** distributions  $k_{\max} = k_{\min} N^{\frac{1}{\gamma-1}}$ 
  - $k_{\max}$  increases quite significantly with  $N$
  - the lower the exponent of the distribution ( $\gamma$ ) the higher the rate of increase

dove  $k_{\max}$  e  $k_{\min}$  sono i gradi massimo e minimo dei nodi della rete. Per distribuzioni a coda leggera la differenza tra il grado massimo e minimo della rete ha un andamento che è logaritmico con il numero di

nodi della rete. Il logaritmo di N ha un andamento che cresce molto poco dunque su una rete heavy tail anche di un miliardo di nodi, la differenza che ci possiamo aspettare di vedere tra i nodi a grado massimo e i nodi a grado minimo è dell'ordine di qualche unità, dunque anche se un nodo ha grado più grande di tutti nella rete, non ha un grado molto più grande rispetto ai nodi con pochi link.

Invece per reti heavy tailed la relazione che c'è tra  $k_{\max}$  e  $k_{\min}$  è una funzione polinomiale di N, il cui polinomio di N dipende da  $\gamma$ .

## Degree distribution and degree variability

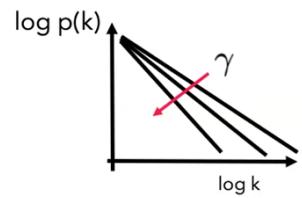


In reti randomiche che sono reti a coda leggera vediamo che l'andamento del  $k_{\max}$  è logaritmico rispetto alla dimensione della rete (N). Nelle reti SCALE-FREE che sono quelle a coda pesante il  $k_{\max}$  ha un andamento crescente con N con una funzione polinomiale di N.

## From heavy tailed to light tail behaviour

---

- Are all power-law distribution heavy tailed?
  - NO. It depends on  $\gamma$
- $\gamma$  controls the **variability** of the degree distribution
- $\gamma < 2$ : Extreme variability
  - “Anomalous” regime
- $2 < \gamma < 3$ : Extreme variability
  - Typical “heavy-tailed” regime
- $\gamma > 3$ : Lower variability
  - More and more similar to a “light-tailed” regime



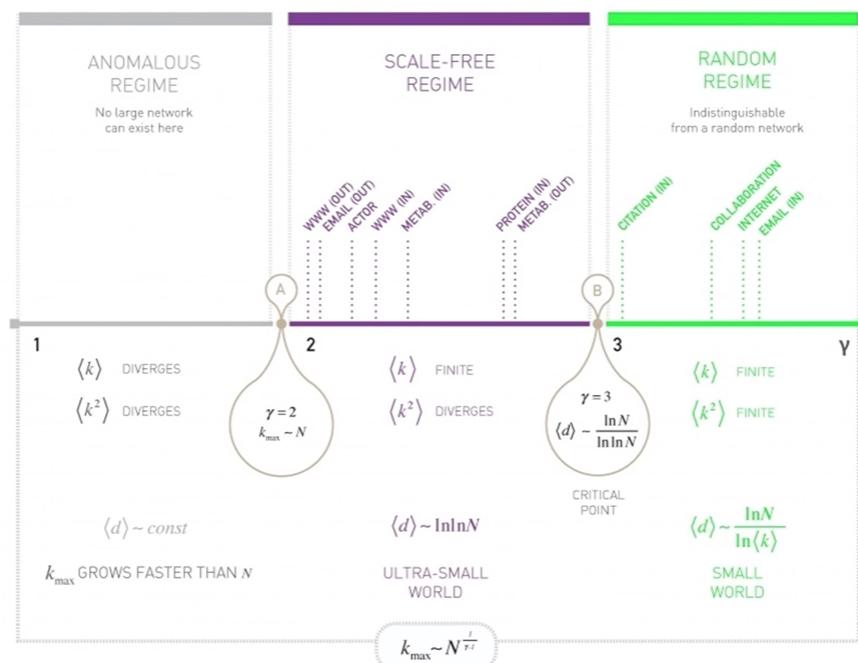
La variabilità la controlla il parametro  $\gamma$  per le reti a coda pesante, in quanto più piccola è  $\gamma$  e più pesante è la coda e dunque maggiore è la probabilità di avere nodi a grado alto e dunque maggiore è il  $k_{\max}$  che possiamo trovare.

E' vero sempre che le distribuzioni a coda pesante sono quelle a power law? Matematicamente SI ma da un punto di vista pratico non sempre è così in quanto la cosa fondamentale per dire che ci troviamo nel caso di una coda pesante è quanto il grado degli hub è grande rispetto a quelli a grado normale o minimo:

---

## From heavy tailed to light tail behaviour

---



A seconda del valore di  $\gamma$  si distinguono 3 regioni.

$\gamma$  deve essere maggiore di 1 per motivi di trattabilità matematica di questi strumenti, fin tanto che  $\gamma$  è

tra 1 e 2 la coda è molto pesante, ci sono degli hub estremamente pesanti e ci troviamo in regime **Anomalo** in quanto la rete è così variabile in termini di distribuzione del grado che si possono sviluppare hub di proporzioni gigantesche e anche questa regione si tratta male. Da un punto di vista fisico reti con  $\gamma$  minore di 2 si trovano in natura ma si descrivono male matematicamente in quanto hanno una grande variabilità del grado e fanno saltare una serie di proprietà matematiche di questi strumenti.

La regione per  $\gamma$  tra 2 e 3 è una regione di grande variabilità, un pò meno del caso  $\gamma$  minore di 2, sempre a coda pesante, ma in questa regione essendo  $\gamma$  leggermente più grande, le aberrazioni matematiche che si trovavano nel caso  $\gamma$  minore di 2 non ci sono più. Si tratta del caso paradigmatico del regime a coda pesante.

Per  $\gamma > 3$  da un punto di vista di modellazione queste sono distribuzioni ancora descritte da una power law ma la coda si allegerisce così tanto che si tratta di reti a coda leggera dal punto di vista del loro comportamento fisico.

Vedendo delle reti reali:

## What regime for real networks?

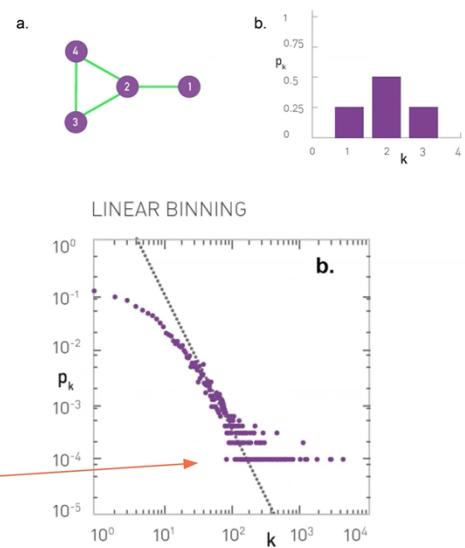
	network	type	$n$	$m$	$z$	$\ell$	$\alpha$	$C^{(1)}$	$C^{(2)}$	$r$	Ref(s).
social	film actors	undirected	449 913	25 516 482	113.43	3.48	2.3	0.20	0.78	0.208	<a href="#">20, 416</a>
	company directors	undirected	7 673	55 392	14.44	4.60	—	0.59	0.88	0.276	<a href="#">105, 323</a>
	math coauthorship	undirected	253 339	496 489	3.92	7.57	—	0.15	0.34	0.120	<a href="#">107, 182</a>
	physics coauthorship	undirected	52 909	245 300	9.27	6.19	—	0.45	0.56	0.363	<a href="#">311, 313</a>
	biology coauthorship	undirected	1 520 251	11 803 064	15.53	4.92	—	0.088	0.60	0.127	<a href="#">311, 313</a>
	telephone call graph	undirected	47 000 000	80 000 000	3.16	—	2.1	—	—	—	<a href="#">8, 9</a>
	email messages	directed	59 912	86 300	1.44	4.95	1.5/2.0	—	0.16	—	<a href="#">136</a>
	email address books	directed	16 881	57 029	3.38	5.22	—	0.17	0.13	0.092	<a href="#">321</a>
	student relationships	undirected	573	477	1.66	16.01	—	0.005	0.001	-0.029	<a href="#">45</a>
	sexual contacts	undirected	2 810	—	—	—	3.2	—	—	—	<a href="#">265, 266</a>
information	WWW <a href="#">nd.edu</a>	directed	269 504	1 497 135	5.55	11.27	2.1/2.4	0.11	0.29	-0.067	<a href="#">14, 34</a>
	WWW Altavista	directed	203 549 046	2 130 000 000	10.46	16.18	2.1/2.7	—	—	—	<a href="#">74</a>
	citation network	directed	783 339	6 716 198	8.57	—	3.0/—	—	—	—	<a href="#">351</a>
	Roget's Thesaurus	directed	1 022	5 103	4.99	4.87	—	0.13	0.15	0.157	<a href="#">244</a>
	word co-occurrence	undirected	460 902	17 000 000	70.13	—	2.7	—	0.44	—	<a href="#">119, 157</a>
technological	Internet	undirected	10 697	31 992	5.98	3.31	2.5	0.035	0.39	-0.189	<a href="#">86, 148</a>
	power grid	undirected	4 941	6 594	2.67	18.99	—	0.10	0.080	-0.003	<a href="#">416</a>
	train routes	undirected	587	19 603	66.79	2.16	—	—	0.69	-0.033	<a href="#">366</a>
	software packages	directed	1 439	1 723	1.20	2.42	1.6/1.4	0.070	0.082	-0.016	<a href="#">318</a>
	software classes	directed	1 377	2 213	1.61	1.51	—	0.033	0.012	-0.119	<a href="#">395</a>
	electronic circuits	undirected	24 097	53 248	4.34	11.05	3.0	0.010	0.030	-0.154	<a href="#">155</a>
biological	peer-to-peer network	undirected	880	1 296	1.47	4.28	2.1	0.012	0.011	-0.366	<a href="#">6, 354</a>
	metabolic network	undirected	765	3 686	9.64	2.56	2.2	0.090	0.67	-0.240	<a href="#">214</a>
	protein interactions	undirected	2 115	2 240	2.12	6.80	2.4	0.072	0.071	-0.156	<a href="#">212</a>
	marine food web	directed	135	598	4.43	2.05	—	0.16	0.23	-0.263	<a href="#">204</a>
	freshwater food web	directed	92	997	10.84	1.90	—	0.20	0.087	-0.326	<a href="#">272</a>
	neural network	directed	307	2 359	7.68	3.97	—	0.18	0.28	-0.226	<a href="#">416, 421</a>

TABLE II Basic statistics for a number of published networks. The properties measured are: type of graph, directed or undirected; total number of vertices  $n$ ; total number of edges  $m$ ; mean degree  $z$ ; mean vertex–vertex distance  $\ell$ ; exponent  $\alpha$  of degree distribution if the distribution follows a power law (or  $-\alpha$  if not; in/out-degree exponents are given for directed graphs); clustering coefficient  $C^{(1)}$  from Eq. (3); clustering coefficient  $C^{(2)}$  from Eq. (6); and degree correlation coefficient  $r$ , Sec. III F. The last column gives the citation(s) for the network in the bibliography. Blank entries indicate unavailable data.

questa è una tabella famosa in cui gli autori prendono una serie di reti e ne hanno caratterizzato secondo questi indici. In questa tabella  $\alpha$  è l'equivalente di  $\gamma$ , ovvero l'esponente che si trova nel caso in cui quella rete abbia un buon fitting con una distribuzione power law. La cosa interessante è che diversi reti sono power law ovvero a coda pesante e inoltre che il valore di  $\gamma$  varia abbastanza ma varia nel range tra 2 e 3.

## CCDF vs Density

- ❑ Density is affected by binning
- ❑ Linear binning: all bins have the **same size**
  - ❑ E.g., steps of 1 degree increment
- ❑ **Low** degrees
  - ❑ All good, many nodes have those degrees
  - ❑  $p_k$  is a reliable estimate of the real value
- ❑ **High** degrees
  - ❑ Only a few nodes have high degrees
    - ❑ Typically, 1 or 0 for each degree value
  - ❑ Plateau for high values of  $k$
  - ❑ Unreliable estimate of the real density



Quando abbiamo parlato di calcolare la densità empirica, si tratta di un calcolo corretto ma va definito il **binning**. Ovvero considero incrementi pari a 1 nella progressione dei gradi che vado a considerare.

Ovvero per  $k = 1$  calcolo quanti nodi ho con tale grado, per  $k = 2$  calcolo quanti nodi ho con tale grado, fino al grado massimo e mi calcolo la frequenza con cui trovo nodi di ciascuno di questi gradi.

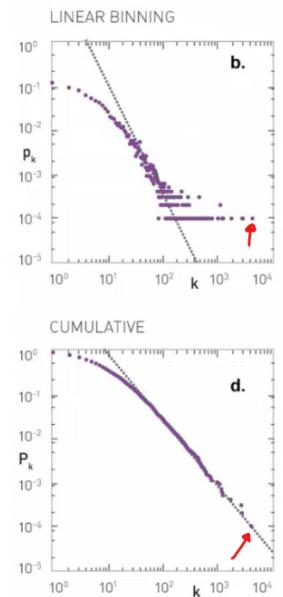
⚠ Il problema di questa metodologia è che in generale in una rete reale avremo tanti nodi a gradi bassi ma pochi a gradi alti. Se vogliamo stimare la probabilità di trovare nodi a grado alto questa sarà molto bassa. Se facciamo su una rete grande un plot della frequenza con cui troviamo nodi di un certo grado, verso i gradi alti appaiono dei plateau come segnato nella slide, abbiamo un solo nodo con quel grado su una rete di 10000 nodi. E' un aberrazione statistica basata sul fatto che stiamo calcolando questa statistica su un campione soltanto o pochissimi campioni. Per ovviare a questo problema non si usa la densità empirica ma si usa una stima della

**Complementary Cumulative Distribution Function** ovvero la probabilità di avere nodi con un certo grado maggiore o uguale di  $k$  che è uguale alla somma, all'evento congiunto, delle probabilità di avere nodi aventi grado  $q$  con  $q$  che va da 1 all'infinito. Ad esempio se vogliamo capire con quale frequenza troviamo nodi con grado maggiore di 3, la probabilità di avere nodi maggiore o uguale di 3 significa avere nodi di grado 4,5,6...

Questa è una statistica più affidabile in quanto i nodi con grado alto è vero che sono pochi, ma se ci chiediamo quale è la probabilità di avere nodi con grado maggiore o uguale a un certo valore è come se stessimo accorpando insieme tutti i nodi che sono in un range molto grande e li mettiamo insieme e dunque la statistica è più affidabile in quanto calcolata su un numero di campioni estremamente più grande.

## Key properties of CCDF

- ❑ Density power law  $\Leftrightarrow$  CCDF power law
  - Density power law with  $\gamma \Leftrightarrow$  CCDF power law with  $(\gamma-1)$
- ❑ Avoids the problems of linear bidding
  - More reliable estimate of the distribution for large values of  $k$
- ❑ ... always use CCDF (and never use density) in analysis



Nell'immagine mostrata sopra si vede la stessa rete in cui viene plottato nel grafico b. la densità campionaria ovvero la probabilità di avere nodi con esattamente il grado  $k$ , in basso invece (grafico d.) abbiamo la CCDF ovvero la probabilità di avere nodi con grado maggiore o uguale di  $k$ . Chiaramente nell'ultimo punto (indicato in rosso) i due valori sono equivalenti ma prima dell'ultimo punto, la seconda curva scende più regolarmente in quanto questo effetto per cui i nodi a grado alto sono pochi, viene ammortizzato dal fatto che mettiamo tutti questi nodi insieme nel calcolo della statistica.

Per quanto riguarda le distribuzioni power law la cui densità è una power law, anche la CCDF sarà una power law con l'unica differenza che cambia l'esponente ovvero sarà  $\gamma-1$ .

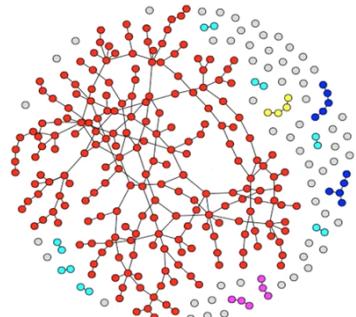
Tutto quello raccontato fin ora va bene ma i dati che si considerano quando si analizza il grado di una rete non sono mai la densità empirica ma è la CCDF empirica. Dunque bisogna calcolarsi la CCDF dei gradi calcolati, fittare la curva della CCDF con una curva in questo caso è log x log e questi dati si allineano su una retta dunque approssimabili con una power law. Ci viene fuori un coefficiente e quello è il  $\gamma$  stimato della nostra distribuzione. Fisicamente non cambia nulla, questo è solo un trucco matematico fondamentale per avere una stima più attendibile del coefficiente  $\gamma$ .

## Giant component

## Giant Component

---

- ❑ In general, a graph can contain a number of **disconnected** components
- ❑ Often there is a very large fraction of nodes that belong to a single “giant” component
- ❑ Giant component
  - ❑ All nodes inside are **connected**
    - ❑ A **finite-length path** exists that joins any two nodes
  - ❑ The size of the giant component is **much (much) larger** than the size of the second-largest component
- ❑ Very often, analysis is done **on the giant component only**
  - ❑ Which anyway includes the vast majority of the nodes



Quando analizziamo una rete abbiamo a che fare con qualcosa di simile a quello mostrato in figura. I **componenti connessi** sono definiti come gruppi di nodi tali per cui è possibile stabilire un path di lunghezza finita tra un nodo e l'altro. Il componente rosso può essere navigato e possiamo raggiungere con un numero finito di passi un qualsiasi altro componente. Non possiamo passare però dal componente giallo a un nodo del componente rosso. Infine ci sono un pulviscolo di nodi (rappresentati in grigio nell'immagine) che sono disconnessi dalla rete, ad esempio nel caso delle reti sociali si tratta di account Twitter attivati e mai usati. Una rete a larga scala ha tanti componenti divisi e nodi disconnessi. **Normalmente un componente è molto più grande di altri**, ad esempio quello rosso in figura. Questo si chiama **GIANT COMPONENT**. Normalmente quando analizziamo un sistema a rete, è interessante solo se ha un giant. Se la rete ha centinaia di migliaia di nodi ma senza giant la rete non è interessante perché il vero valore della rete non è dato dal numero di nodi, ma da come questi sono connessi. Per fortuna, nei sistemi reali è quasi sempre così ovvero esiste un giant component che racchiude più del 90% della rete.

! Il vero interesse sta nell'analisi del giant component, dunque i discorsi della distribuzione del grado fatti in precedenza si applicano principalmente sul giant component.

Esistono delle analisi matematiche anche sul giant component per capire come la sua forma dipende dal grado medio, dunque il grado della rete determina questo:

## Giant Component and Degree Distribution

---

- ❑ Giant Component exists  $\Leftrightarrow \langle k \rangle \geq 1$ 
  - ❑ Every node is **expected** to be connected to at least another node
- ❑ Sizes of the GC
  - ❑  $\langle k \rangle < 1$  sub-critical regime,
    - ❑ Many small clusters of small size,  $N_G \sim \ln N$
  - ❑  $\langle k \rangle > 1$  super-critical regime,
    - ❑ One big component, which is a significant fraction of the entire network ( $N$ )
    - ❑ The rest is spread in many very small components
  - ❑  $\langle k \rangle \gg 1$  ( $\langle k \rangle \sim \ln N$ ) connected regime
    - ❑ One giant component including the entire network  $N_G \sim N$

Esiste un teorema che afferma che:

*esiste un giant component se e solo se il grado medio della rete  $\langle k \rangle \geq 1$ .*

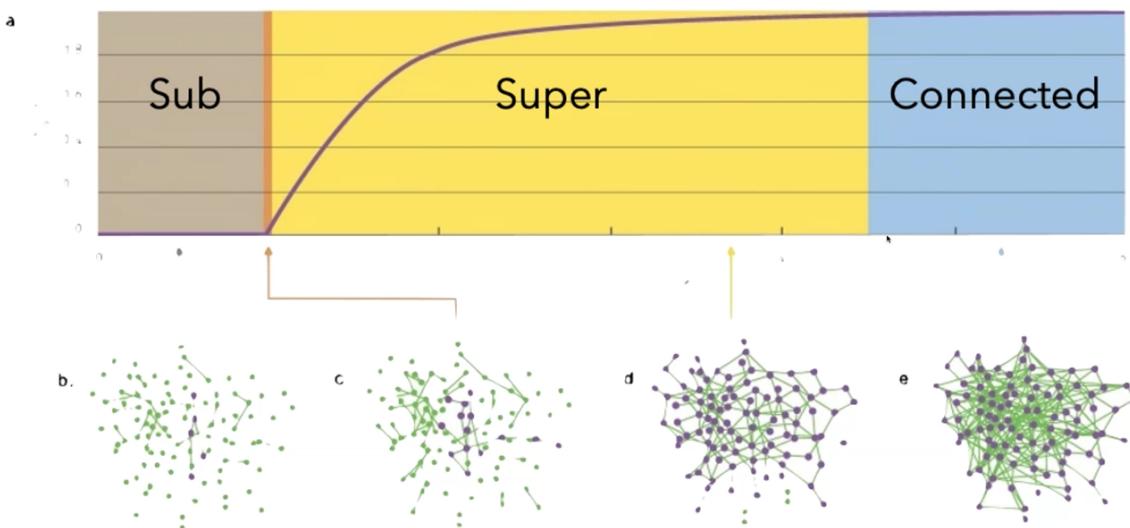
Il nodo medio ha un grado  $\geq 1$ , ovvero mediamente ogni nodo della rete ci si aspetta che abbia un grado  $\geq 1$  ovvero che sia connesso con almeno un altro nodo. Questo è molto intuitivo in quanto deve esistere una grande maggioranza di nodi per cui esiste un link per cui è possibile andare da un componente a un altro in numero finito di passi, esplorando così tutti i componenti della rete.

Ragionando però sul grado medio questo non esclude che ci possano essere dei nodi isolati. La condizione necessaria è sufficiente di  $\langle k \rangle \geq 1$  vuol dire che mediamente possiamo spostarci ma non vuol dire che non ci siano nodi isolati. Al variare del grado medio di  $\langle k \rangle$  si caratterizza anche la forma del giant component:

- Se  $\langle k \rangle < 1$  il giant component non esiste e dunque siamo nella zona che non è interessante, nel SUB-CRITICAL REGIME, ovvero la rete è organizzata in tanti cluster la cui dimensione è logaritmica con il numero di nodi.
- Se  $\langle k \rangle > 1$  siamo nella SUPER-CRITICAL REGIME. Abbiamo un solo componente con un pulviscolo di nodi non collegati che è il caso tipico visto in figura precedentemente.
- Se  $\langle k \rangle \gg 1$  ovvero via via che cresce il grado medio vuol dire che ho una rete più fitta in termini di link, fino al logaritmo del numero di nodi vuol dire che si arriva al CONNECTED REGIME ovvero la rete diventerebbe tutta costituita da componenti rossi. In questo caso **SICURAMENTE** non abbiamo nodi isolati.

In tutti questi casi l'analisi si fa solo sul giant component.

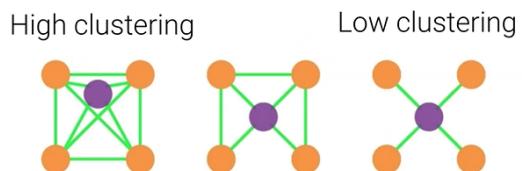
## Giant Component and Degree Distribution



## Clustering

### Clustering concept

- ❑ Cluster = groups of users **very well connected** to each other
  - ❑ In social networks, closed communities of people, all friends of each other
- ❑ High clustering = high **cohesion** of the network
  - ❑ High resilience
  - ❑ Quick diffusion of information
  - ❑ ...

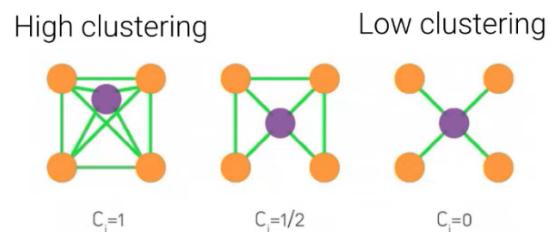


Il cluster è un concetto non definito in modo rigoroso ma intuitivamente permette di capire quanto i nodi di una rete sono collegati tra di loro.

Il cluster e i componenti connessi sono differenti in quanto in queste 3 piccole reti esiste un unico componente连通 ma la struttura dei cluster è radicalmente diversa tra di loro.

## Clustering coefficient

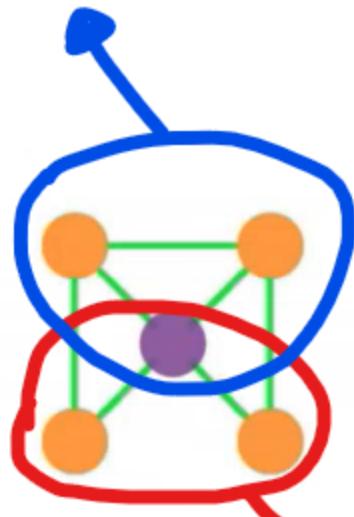
- ❑ A **measure** of how much clustered a network is
- ❑ Typically, expressed with a numerical index taking values from 0 to 1
  - ❑ The higher the index, the more the network is clustered
- ❑ For any such index
  - ❑ Hub network has clustering = 0
  - ❑ Full mesh (clique) has clustering = 1



Il clustering si misura con un indice  $C_i$  compreso tra 0 e 1. Con  $C_i = 0$  abbiamo una condizione di basso clustering mentre con  $C_i = 1$  abbiamo una full mesh della nostra rete, dove tutti i nodi sono collegati tra di loro direttamente. La rete ad hub è fatta con un unico nodo centrale in cui tutti i nodi sono connessi a lui ha  $C_i = 0$ .

I modi per calcolare gli indici si basano sul concetto di **triangolo** e **tripletta**:

# TRIANGolo



$$C_i = 1/2$$

# TRIPLETTA

Un triangolo si tratta di 3 link ovvero 3 nodi connessi da 3 link. Un triangolo è anche una tripletta

Un triangolo dunque è per forza anche una tripletta, mentre una tripletta è un triangolo mancato, ovvero solo 2 link, ovvero un nodo collegato ad altri 2.

Per misurare il clustering si calcolano i triangoli e le triplette e si contano quanti triangoli ci sono centrati su ciascuno dei nodi. Il rapporto mi da il coefficiente di clustering. Questo indice mi misura quel fenomeno, in quanto nella full mesh ci sono tutti i link possibili, dunque per ogni tripletta ho sempre anche un triangolo:

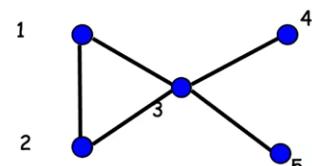
## Example of $C^{(1)}$ computation

### □ Triangles

- 3 in total (one centred at each node: 1, 2, 3)

### □ Triplets

- none centred at nodes 4, 5
- node 1 – 213
- node 2 – 123
- node 3 – 134, 135, 234, 235, 132, 435



$$C^{(1)} = \frac{3}{1+1+6} = \frac{3}{8}$$

In questo esempio abbiamo 3 triangoli, anche se potremmo obiettare che si tratta dello stesso triangolo centrato rispettivamente su 1,2 e 3. Geometricamente l'obiezione è corretta ma per come abbiamo descritto questo indice di clustering va preso ogni nodo e verificato quanti triangoli ci sono.

## Local clustering coefficient ( $C_i$ ) and its average value ( $C^{(2)}$ )

### □ Local clustering coefficient of node $i$ , $C_i$

- How much the local neighborhood of  $i$  is clustered

$$C_i = \frac{\text{triangles centered at node } i}{\text{triples centered at node } i}$$

### □ $C^{(2)}$ = Average of the local clustering coefficients

- The mean of the ratios

$$C^{(2)} = \frac{\sum_{i=1}^N C_i}{N}$$

Esiste un altro indice con lo stesso ruolo, chiamato  $C^2$ . Qui ho una visione locale per uno specifico nodo. Dunque prendo un nodo  $i$ -esimo, conto quanti triangoli ci sono centrati su quel nodo diviso il numero di triplette centrate su quel nodo. Questo lo faccio per tutti i nodi e me ne calcolo la **MEDIA ARITMETICA**.

**I valori di  $C^1$  e  $C^2$  saranno sempre diversi.** L'indice  $C^2$  è in modo globale il valor medio, ma una volta calcolato l'indice abbiamo il valore  $C_i$  di tutti i nodi. Per come abbiamo introdotto i triangoli e le triplette, questo  $C_i$  preso sul singolo nodo mi da un idea del coefficiente di clustering locale di quel nodo e non del coefficiente di clustering globale della rete.

## Clustering coefficients of real networks

	network	type	n	m	z	$\ell$	$\alpha$	$C^{(1)}$	$C^{(2)}$	r	Ref(s.)
social	film actors	undirected	449 913	25 516 482	113.43	3.48	2.3	0.20	0.78	0.208	20, 416
	company directors	undirected	7 673	55 392	14.44	4.60	—	0.59	0.88	0.276	105, 323
	math coauthorship	undirected	253 339	496 489	3.92	7.57	—	0.15	0.34	0.120	107, 182
	physics coauthorship	undirected	52 909	245 300	9.27	6.19	—	0.45	0.56	0.363	311, 313
	biology coauthorship	undirected	1 520 251	11 803 064	15.53	4.92	—	0.088	0.60	0.127	311, 313
	telephone call graph	undirected	47 000 000	80 000 000	3.16	—	2.1	—	—	—	8, 9
	email messages	directed	59 912	86 300	1.44	4.95	1.5/2.0	—	0.16	—	136
	email address books	directed	16 881	57 029	3.38	5.22	—	0.17	0.13	0.092	321
sexual contacts	student relationships	undirected	573	477	1.66	16.01	—	—	0.005	0.001	—0.029
	undirected	2 810	—	—	—	3.2	—	—	—	—	265, 266
information	WWW nd.edu	directed	269 504	1 497 135	5.55	11.27	2.1/2.4	0.11	0.29	—0.067	14, 34
	WWW Altavista citation network	directed	203 549 046	2 130 000 000	10.46	16.18	2.1/2.7	—	—	—	74
	Roget's Thesaurus word co-occurrence	directed	783 339	6 716 198	8.57	—	3.0/—	—	—	—	351
	Internet	undirected	1 022	5 103	4.99	4.87	—	0.13	0.15	0.157	244
	power grid	undirected	460 902	17 000 000	70.13	—	2.7	—	0.44	—	119, 157
technological	train routes	undirected	4 941	6 594	2.67	18.99	—	0.10	0.080	—0.003	416
	software packages	undirected	587	19 603	66.79	2.16	—	—	0.69	—0.033	366
	software classes	directed	1 439	1 723	1.20	2.42	1.6/1.4	0.070	0.082	—0.016	318
	electronic circuits	undirected	1 377	2 213	1.61	1.51	—	0.033	0.012	—0.119	395
	peer-to-peer network	undirected	24 097	53 248	4.34	11.05	3.0	0.010	0.030	—0.154	155
biological	metabolic network	undirected	880	1 296	1.47	4.28	2.1	0.012	0.011	—0.366	6, 354
	protein interactions	undirected	765	3 686	9.64	2.56	2.2	0.090	0.67	—0.240	214
	marine food web	directed	2 115	2 240	2.12	6.80	2.4	0.072	0.071	—0.156	212
	freshwater food web	directed	135	598	4.43	2.05	—	0.16	0.23	—0.263	204
	neural network	directed	92	997	10.84	1.90	—	0.20	0.087	—0.326	272
	undirected	307	2 359	7.68	3.97	—	—	0.18	0.28	—0.226	416, 421

Are these coefficients large or small?

TABLE II Basic statistics for a number of published networks. The properties measured are: type of graph, directed or undirected; total number of vertices  $n$ ; total number of edges  $m$ ; mean degree  $z$ ; mean vertex–vertex distance  $\ell$ ; exponent  $\alpha$  of degree distribution if the distribution follows a power law (or “—” if not; in/out-degree exponents are given for directed graphs); clustering coefficient  $C^{(1)}$  from Eq. (3); clustering coefficient  $C^{(2)}$  from Eq. (6); and degree correlation coefficient  $r$ . Sec. III E The last column gives the citation(s) for the network in the bibliography. Blank entries indicate unavailable data.

In una rete molto grande è molto più improbabile che una grande frazione di nodi siano connessi tra di loro rispetto al caso di una rete più piccola. Da solo il numero assoluto del clustering racconta solo una parte della storia.

? I nodi a grado alto ci aspettiamo che abbiamo un coefficiente di clustering locale  $C_i$  più alto o più basso dei nodi a grado basso?

Se pensiamo in una rete sociale a personaggi famosi come Obama, loro hanno tantissimi follower ma non tutte le persone che lo seguono sono collegate tra di loro, proprio perché essendo tante non è detto che siano collegate tra di loro e la probabilità si abbassa di molto.

Il coefficiente di clustering va messo in relazione al grado dei nodi o alla dimensione della rete. Si usa una metodologia per confrontare le reti tra di loro. Come si studiano le reti in realtà?

Quando studiamo una rete abbiamo un dataset che descrive i nodi e i suoi collegamenti, sulla base di questi dati calcoliamo la degree distribution e poi cerchiamo di capire e astrarre per capire le proprietà strutturali della rete. Quando passiamo ai modelli astratti, è possibile confrontare il modello della rete con modelli di cui si sa tutto, ovvero con modelli di riferimento.

## Comparison of networks

---

- Methodology to study real networks
  - Measure some indices from the data (or by fitting them, e.g., for the degree)
  - Compare what you get with **abstract models** of networks
- Why?
  - Abstract models are **paradigmatic** of certain known properties
  - If your network “is similar” to one of them, you can expect it to behave as that specific model
- Note
  - The same network can be **similar to more than one model**, for different indices
    - E.g., similar to “model A” for the degree distribution and to “model B” for the clustering coefficient

Quando dobbiamo confrontare una rete power law con una rete a comportamento esponenziale dobbiamo essere sicuri di confrontare reti a parità di nodi e di link dunque questi confronti tra modelli di rete si fanno sempre a parità di numero di nodi e di link. Se da una rete come quella di Facebook con miliardi e miliardi di nodi e link volessimo derivarne un modello astratto, dovremmo derivare un modello che ha lo stesso numero di nodi e lo stesso numero di link della rete di Facebook. Il numero di nodi mi dice quanto è grande la rete mentre il numero di link mi dice quanto valore c'è nella rete o meglio quanto costa costruirlo nella rete (si ripensi all'esempio di costruire un link nella rete internet cablato con fibra ottica).

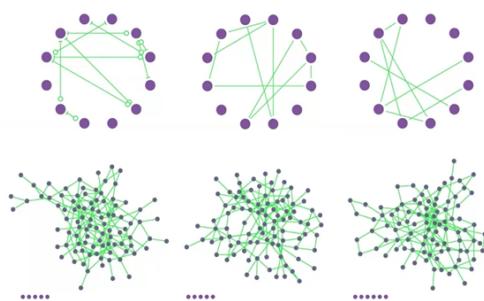
Fra i modelli di riferimento, uno molto famoso e utile è quello di Erdos-Renyi.

## Random Network model

### “Random Network” model: the Erdos-Renyi graph

---

- Very simple model
  - $N$  nodes
  - each possible link exists with **probability**  $p$
- Consequences
  - If you generate two networks with the same parameters, you will obtain two **different** networks
  - But, the **structural properties** will be the same



è stato il primo modello di questo tipo proposto, ed è detto random network. Ci sono un numero di nodi

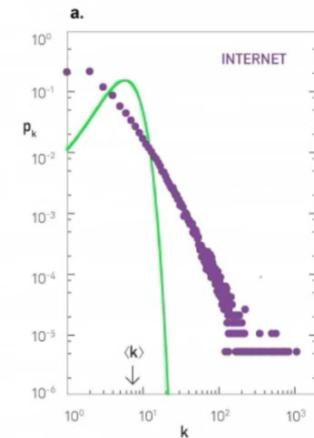
e tutti i link fra i nodi vengono messi con probabilità  $p$ .

Se nomino i nodi da 1 a 10, per decidere se creare un link tra il nodo 1 e 2 uso l'estrazione di una variabile aleatoria, con probabilità  $p$  aggiungo il link, con probabilità  $1-p$  non lo aggiungo. Questo modo di costruire le reti sembra aleatorio, se con lo stesso numero di nodi  $N$  e lo stesso parametro  $p$  costruisco  $n$  volte le reti, ottengo in generale reti diverse anche se hanno lo stesso numero di nodi.

Queste reti hanno una proprietà strutturale in comune, ovvero la probabilità di avere una connessione diretta tra qualunque coppia di nodi è la stessa. Funziona tutto intuitivamente in quanto questi strumenti si usano per analizzare reti grosse, ovvero nel limite del teorema centrale, ovvero le differenze minuscole che derivano dal fatto che ci sia un aleatorietà nel processo di costruzione spariscono perché poi si considerano indici complessivi. Ovvero quando analizzo questi sistemi grandi a grafo, non sono mai interessato alla proprietà del nodo  $x$  che sta nella zona  $y$  della rete, ma sono interessato a indici complessivi che mi descrivono la rete nel complesso. In particolare una rete costruita in questo modo ha una degree distribution a coda leggera di Poisson e ha un coefficiente di clustering basso:

### Key properties of Erdos-Renyi graphs

- Average degree  $\langle k \rangle = p(N-1) \sim pN$  (as  $N$  is large)
  - As each node can be connected with any other nodes ( $N-1$ ) with probability  $p$
- Degree distribution is approximately Poisson
  - Tail decays **very fast**
- Clustering coefficient is  $p \sim \langle k \rangle / N$ 
  - As the probability that a triplet is "closed" is equal to  $p$
- For a given average degree  $\langle k \rangle$ ,  
the clustering coefficient is **proportional to  $1/N$** 
  - The larger the network, the less it is clustered



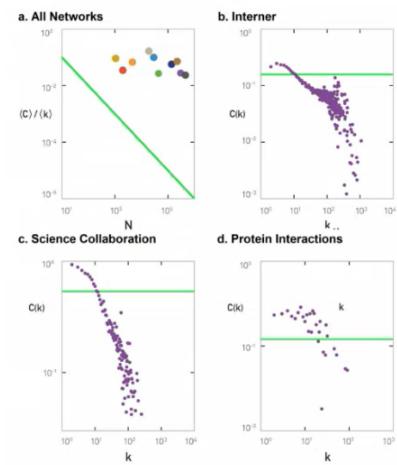
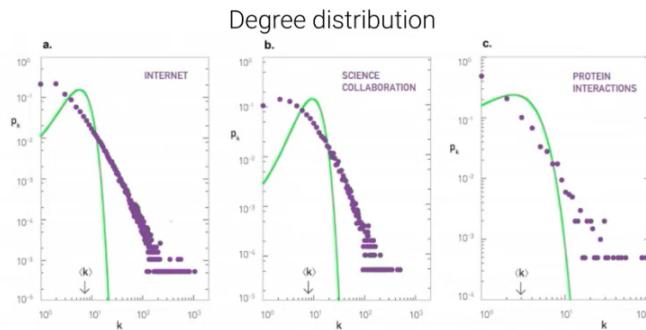
Il grado medio di questa rete è uguale a  $p(N-1)$  o approssimabile per  $pN$ . Perchè il grado medio che vuol dire il numero medio di link che ogni nodo ha è uguale a  $p \times N$ ?

Quanti link mi aspetto che esistano per il nodo generico che prendo? Il nodo medio potenzialmente potrebbe avere  $N-1$  link in quanto potenzialmente potrebbe essere connesso con tutti gli altri nodi, ma ciascuno di questi link esiste con probabilità  $p$ . Dato che  $N$  normalmente è molto grande il grado medio si approssima bene con  $p \times N$ . In tutte queste reti costruite in questo modo, la degree distribution decade molto velocemente. Il clustering coefficient è approssimativamente uguale a  $p$ . Lo abbiamo definito come la probabilità che una tripletta in realtà sia un triangolo. Ovvero dato un nodo che è collegato ad altri 2 nodi, questi altri 2 nodi siano collegati tra di loro, ovvero che tra questi 2 nodi esista un link. Ma in una rete Erdos-Renyi la probabilità che qualunque link esista è uguale a  $p$  e questo spiega come mai il clustering coefficient si dimostra essere approssimativamente uguale a  $p$ . La cosa interessante è che vista la prima proprietà per cui il grado medio è uguale a  $p \times N$ , il clustering coefficient è approssimativamente uguale a  $\langle k \rangle / N$ . Il  $\langle k \rangle$  di una rete al variare della dimensione di una rete non cresce molto, quindi nelle reti di Erdos-Renyi il coefficiente di clustering è proporzionale

a  $1/N$ . Dunque più una rete è grande, più il coefficiente di clustering diminuisce e questo per quello detto prima non sorprende, ma in questo tipo di rete scende molto velocemente. **Le reti di Erdos-Renyi sono reti poco coese.**

## Realism of the ER model

- ❑ Networks are **very hardly similar** to ER graphs
  - ❑ Light tail vs **heavy** tail
  - ❑ Local clustering typically **decreases** with the degree
    - ❑ Which is **intuitive**



E' difficile trovare delle reti che hanno l'andamento di Erdos-Renyi nella realtà. Si tratta di un modello matematico molto semplice ma è un modello di riferimento per capire il clustering delle reti di interesse.

**?** Come capiamo se il coefficiente di clustering di una rete è alto o basso?

Si confronta il valore con il coefficiente di clustering della rete di Erdos-Renyi equivalente alla rete di clustering che sto studiando. Se ho una rete che ha un certo numero di nodi e di link. Mi costruisco una rete equivalente alla mia (stesso numero di nodi e di link) che però è costruita secondo Erdos-Renyi. Se anche la mia distribuzione come quella di ER ha una distribuzione a coda leggera posso dire che da un punto di vista del grado la mia rete si comporta come quella di ER. Quello per cui si usa spesso ER è come parametro di confronto per il clustering coefficient.

Nel riquadro sulla destra vediamo il clustering coefficient delle reti equivalenti a quelle che sto analizzando riga per riga, se e se fossero organizzate secondo il modello di ER. Questi coefficienti di clustering delle reti di ER vanno come  $1/n$ , con  $n$  numero di nodi. La colonna  $z$  è l'equivalente del grado medio. Il coefficiente di clustering delle reti equivalenti secondo il modello di ER va come  $1/n$ .

## Usefulness of ER graphs

- So, why do we bother?
- We use them only as a **reference** point
  - In particular for assessing how clustered a network is
- $C^{(1)}$  and  $C^{(2)}$  **alone** are not particularly useful
  - As they must be considered with respect to the **size** of the network,  $N$
- Compare  $C^{(1)}$  and  $C^{(2)}$  with the ones of the **equivalent ER graph**
  - To understand how **far** (or **close**) our network is to a very low clustered one

Dunque calcoliamo il clustering coefficient  $C$  measured della rete di interesse (potrebbe essere  $C^1$  o  $C^2$  non è importante), lo confronto con  $1/n$  ovvero il valore atteso della rete ER e confronto questi 2 numeri.

Guardando la tabella, tra le biology collaborations e le power grid quale è più clusterizzata? Quella delle biology collaborations in quanto è più lontana dal modello equivalente di ER o detto in altri termini, è vero che i due coefficienti di clustering misurati sono uguali ma sono ottenuti su due reti che hanno 2 ordini e mezzo di grandezza di differenza in termini di numero di nodi.

27/05/2023

### Part 1. Reading graphs from files / Writing graphs to files

```
In [2]: fg = read("./facebook.ncol", format = "ncol", directed = True)
```

#### Notes

- ncol is a simple format of the kind n1 n2 <weight>
  - the third column is optional, and by default it is considered as the weight of the link, if present
- directed is True by default

Dobbiamo specificare alla funzione `read` se deve interpretare i dati passati come link direzionali o meno. Il formato `.ncol` infatti non prevede una specifica esplicita riguardo la direzionalità del grafo. Nel dubbio i grafi si analizzano come non direzionali, in questa fase lo mettiamo uguale a `True` solo per motivi didattici.

$z = \langle k \rangle$

network	n	z	clustering coefficient $C$ measured	random graph
Internet (autonomous systems) <sup>a</sup>	6 374	3.8	0.24	0.00060
World-Wide Web (sites) <sup>b</sup>	153 127	35.2	0.11	0.00023
power grid <sup>c</sup>	4 941	2.7	0.080	0.00054
biology collaborations <sup>d</sup>	1 520 251	15.5	0.081	0.000010
mathematics collaborations <sup>e</sup>	253 339	3.9	0.15	0.000015
film actor collaborations <sup>f</sup>	449 913	113.4	0.20	0.00025
company directors <sup>g</sup>	7 673	14.4	0.59	0.0019
word co-occurrence <sup>h</sup>	460 902	70.1	0.44	0.00015
neural network <sup>i</sup>	282	14.0	0.28	0.049
metabolic network <sup>j</sup>	315	28.3	0.59	0.090
food web <sup>k</sup>	134	8.7	0.22	0.065

Table 2.1: Number of vertices  $n$ , mean degree  $z$ , and clustering coefficient  $C$  for a number of different networks. Numbers are taken from <sup>a</sup>Pastor-Satorras *et al.* (2001), <sup>b</sup>Adamic (1999), <sup>c</sup>Watts and Strogatz (1998), <sup>d</sup>Newman (2001b), <sup>e</sup>Newman (2001d), <sup>f</sup>Newman *et al.* (2001), <sup>g</sup>Cancho and Solé (2001), <sup>h</sup>Montoya and Solé (2001), <sup>i</sup>Fell and Wagner (2000).

Quelli indicati in verde sono invece il numero di nodi mentre quelli in blu il numero di link:

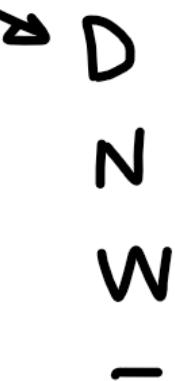
### A summary of the graph:

```
In [3]: summary(fg, verbosity=1, max_rows = 25, edge_list_format = 'edgelist')  
IGRAPH DNW- 45813 264004 --  
+ attr: name (v), weight (e)  
+ edges (vertex names):  
    edge      weight  
[0] 2->3          13  
[1] 2->79         1  
[2] 2->872         1  
[3] 2->1043        1  
[4] 2->1847        8  
[5] 2->3306        11  
[6] 2->3372        3  
[7] 2->4605         1  
[8] 2->5402         4  
[9] 2->5875         4  
[10] 2->8785        16  
[11] 2->10609       25  
[12] 2->10998        1  
[13] 2->11186        5  
[14] 2->12172        3  
[15] 2->17766        1  
[16] 2->18086        6  
[17] 2->18745        1  
[18] 2->30579        5  
[19] 2->42558        1
```

questo vuol dire che il file .ncol deve avere

Ciacuna delle lettere riportate in figura ha un significato preciso:

```
IGRAPH DNW- 45813 264004 --  
+ attr: name (v), weight (e)  
+ edges (vertex names):  
    edge      weight  
[0] 2->3          13  
[1] 2->79         1  
[2] 2->872         1  
[3] 2->1043        1  
[4] 2->1847        8  
[5] 2->3306        11  
[6] 2->3372        3  
[7] 2->4605         1  
[8] 2->5402         4  
[9] 2->5875         4  
[10] 2->8785        16  
[11] 2->10609       25  
[12] 2->10998        1  
[13] 2->11186        5
```



- **D:** ci dice se il grafo è direzionale o meno, se è presente un trattino significa che non è un grafo direzionale
- **N:** named, significa che i nodi all'interno del grafo hanno un attributo testuale. Infatti l'attributo name è associato ai vertici (v) mentre l'attributo weight è associato agli edge (e). Quando importiamo un grafo con igraph, igraph legge riga per riga il file .ncol, nel caso specifico vede 2 -> 3 con peso 13 dunque si crea un primo oggetto nodo a cui da nome 2, poi crea un oggetto nodo con nome 3, e si crea l'oggetto edge che collega 2 a 3 e gli da un attributo 13. Sulla seconda riga, igraph vede che il nodo 2 esiste già, mentre il nodo 79 non esiste e lo crea. Le etichette associate ai nodi sono indifferenti, sarebbero potuti essere numeri così come nomi.

- **W:** weighted, ovvero esiste un attributo weight che appartiene agli edge.

L'oggetto graph ha al suo interno nodi ed edge che organizza come array ovvero all'interno dell'oggetto graph c'è un array di edge e un array di vertici. Vuol dire che l'identificativo univoco di un nodo non è tanto il suo nome, io potrei crearmi un altro nodo all'interno di questo grafo che si chiama 2 ma sarebbe un oggetto nodo distinto da quello che igraph si è creato invocando la read. In generale igraph distingue i nodi e i vertici per l'indice interno con cui li memorizza nell'array.

E' possibile importare il grafo dicendogli che i link non devono essere direzionali. Basta importare con la funzione read il dataset con il `directed = False`:

```
In [4]: fg_u = read("./facebook.ncol", format = "ncol", directed = False)
summary(fg_u, verbosity = 1, max_rows = 25, edge_list_format = "edgelist")
```

```
IGRAPH UNW- 45813 264004 --
+ attr: name (v), weight (e)
+ edges (vertex names):
      edge      weight
[0] 2--3          13
[1] 2--79         1
[2] 2--872        1
[3] 2--1043       1
[4] 2--1847        8
[5] 2--3306       11
[6] 2--3372        3
[7] 2--4605        1
[8] 2--5402        4
[9] 2--5875        4
[10] 2--8785       16
[11] 2--10609      25
[12] 2--10998       1
[13] 2--11186       5
[14] 2--12172       3
[15] 2--17766       1
[16] 2--18086       6
[17] 2--18745       1
[18] 2--30579       5
[19] 2--42558       1
[20] 2--3          8
[21] 5--27         163
[22] 5--108        3
[23] 5--129        2
```

**RELLA** la visualizzazione degli edge è cambiata in quanto non c'è più la freccetta ma i --.

Anche se il grafo adesso è non direzionale continuano ad esserci 2 link tra il nodo 2 e il nodo 3:

```

IGRAPH UNW- 45813 264004 --
+ attr: name (v), weight (e)
+ edges (vertex names):
      edge      weight
[0] 2--3       13
[1] 2--79      1
[2] 2--872     1
[3] 2--1043    1
[4] 2--1847    8
[5] 2--3306    11
[6] 2--3372    3
[7] 2--4605    1
[8] 2--5402    4
[9] 2--5875    4
[10] 2--8785   16
[11] 2--10609  25
[12] 2--10998  1
[13] 2--11186  5
[14] 2--12172  3
[15] 2--17766  1
[16] 2--18086  6
[17] 2--18745  1
[18] 2--30579  5
[19] 2--42558  1
[20] 2--3       8
[21] 5--27     163

```

questo non è un errore ma il motivo è che da un punto di vista di analisi dei grafi è possibile avere più di un link che collega la stessa coppia di nodi. In genere questo è un modo per rappresentare i multi-layer graph.

Per fare alcuni esercizi usiamo un toy graph come molti meno nodi ed edge:

## Part 2. Connected components, Giant Component & Subgraphs

```
In [6]: toy_g = read("./toy_graph_2.ncol", format = "ncol", directed = True)
summary(toy_g, verbosity = 1, edge_list_format = "edgelist")
```

```
IGRAPH DNW- 7 9 --
+ attr: name (v), weight (e)
+ edges (vertex names):
  edge    weight
[0] 1->2    0.100
[1] 1->3    0.300
[2] 1->4    0.010
[3] 1->5    0.500
[4] 2->5    1
[5] 3->4    0.900
[6] 5->a    0.030
[7] a->b    0.020
[8] b->a    0.010
```

è possibile usare la funzione `is_connected` per capire se il grafo su cui chiamiamo la funzione è connesso oppure no.

Il parametro `mode` che la funzione prende ci dice come devono essere considerati i link nel calcolo della funzione. Capire se un grafo è connesso significa capire se partendo da qualunque nodo è possibile arrivare a qualsiasi altro nodo del grafo. Il problema è che se il grafo è direzionale ci sono 2 modi per muoversi nel grafo, dunque possiamo considerare o meno la direzione stabilita dal link. Se la modalità è `WEAK` sto consentendo di muovermi in entrambe le direzioni anche se magari la direzione consentita è in un solo verso.

Provando ad andare a rimuovere il link tra il nodo 5 ed il nodo a:

```
IGRAPH DNW- 7 9 --
+ attr: name (v), weight (e)
+ edges (vertex names):
  edge    weight
[0] 1->2    0.100
[1] 1->3    0.300
[2] 1->4    0.010
[3] 1->5    0.500
[4] 2->5    1
[5] 3->4    0.900
[6] 5->a    0.030
[7] a->b    0.020
[8] b->a    0.010
```

```
# Check whether the graph is connected or not
toy_g.is_connected(mode = "WEAK")
toy_g.delete_edges(toy_g.get_eid("5", "a"))
toy_g.is_connected(mode = "WEAK")
```

sto artificialmente creando 2 componenti connessi infatti non ci sarà modo di passare da un gruppo di nodi all'altro.

La funzione `connected_components` spacchetta il grafo nei suoi componenti connessi:

```
# Compute the connected components in the graph
# - "WEAK" does not consider the direction of edges
toy_g_conn_comp = toy_g.connected_components(mode = "WEAK")

# the number of components
len(toy_g_conn_comp)

# the membership of vertices in the components
toy_g_conn_comp.membership

# the sizes of the components
toy_g_conn_comp.sizes()

# the vertex IDs of the first components
toy_g_conn_comp[0]

# the Giant Componet (the biggest components)
toy_g_GC = toy_g_conn_comp.giant()
summary(toy_g_GC, verbosity = 1, edge_list_format = "edgelist")
```

2

[0, 0, 0, 0, 0, 1, 1]

[5, 2]

[0, 1, 2, 3, 4]

permettendoci di sapere alcune informazioni come ad esempio il numero di componenti del grafo.

La **membership** invece è un vettore che ha la stessa dimensione del numero di nodi del grafo, nell'immagine è un vettore grande 7 in quanto nel grafo ci sono 7 nodi.

La posizione i-esima è associata al nodo in posizione i-esima nel vettore dei nodi dell'oggetto grafo. Il vettore membership ci dice che il primo nodo del grafo appartiene al componente连通的 di indice 0. Mentre 1 è l'indice del secondo componente连通的. Dunque questo vettore di membership mi dice che il nodo 0 fa parte del primo componente连通的, il nodo di indice 1 fa parte del secondo componente连通的 e così via.

La funzione `giant()` prende all'interno dei componenti connessi del grafo quello di dimensione maggiore e lo restituisce come oggetto grafo:

```
# the Giant Component (the biggest components)
toy_g_GC = toy_g_conn_comp.giant()
summary(toy_g_GC, verbosity = 1, edge_list_format = "edgelist")
```

```
Out[8]: 2
Out[8]: [0, 0, 0, 0, 0, 1, 1]
```

```
Out[8]: [5, 2]
```

```
Out[8]: [0, 1, 2, 3, 4]
```

```
IGRAPH DNW- 5 6 --
+ attr: name (v), weight (e)
+ edges (vertex names):
      edge    weight
[0] 1->2    0.100
[1] 1->3    0.300
[2] 1->4    0.010
[3] 1->5    0.500
[4] 2->5    1
[5] 3->4    0.900
```



Sul dataset di Facebook se proviamo a stampare un ordinamento delle dimensioni partendo dal più grande otteniamo i primi 20 componenti connessi in ordine di dimensione:

```
# sizes (sorted, first 20 elements)
sorted(fg_cc.sizes(), reverse=True)[0:19]
```

```
IGRAPH DNW- 43953 262631 --
+ attr: name (v), weight (e)
+ edges (vertex names):
      edge    weight
[0] 2->3    13
[1] 2->79   1
[2] 2->872   1
[3] 2->1043  1
[4] 2->1847  8
[5] 2->3306  11
[6] 2->3372  3
[7] 2->4605  1
[8] 2->5402  4
[9] 2->5875  4
[10] 2->8785  16
[11] 2->10609 25
[12] 2->10998 1
[13] 2->11186 5
[14] 2->12172 3
[15] 2->17766 1
[16] 2->18086 6
[17] 2->18745 1
[18] 2->30579 5
[19] 2->42558 1
```

```
Out[9]: 45813
```

```
Out[9]: 264004
```

```
Out[9]: 842
```

```
Out[9]: [43953, 6, 5, 5, 5, 5, 5, 5, 5, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4]
```

questo è il classico caso di esistenza dei giant component ovvero il componente连通的 più grande di tutti ha una dimensione che è quasi uguale alla dimensione complessiva del grafo nell'ordine del 90% e il secondo componente più grande è immensamente più piccolo (dimensione pari a 6).

## Induced subgraph

La funzione select serve a estrarre dal giant component solo nodi con grado maggiore di 100.

### Induced subgraph

```
In [10]: # Select only nodes with degree > 100 from the FB Giant Component
vs = fb_GC.vs.select(_degree_gt = 100)

# "Induced" subgraph: graph composed by
# - a set of vertices (from the original graph)
# - only the edges connecting those vertices (in the original subgraph)
fb_g_sub = fb_GC.induced_subgraph(vs)

try:
    del visual_style
    visual_style = {}
except NameError:
    visual_style = {}

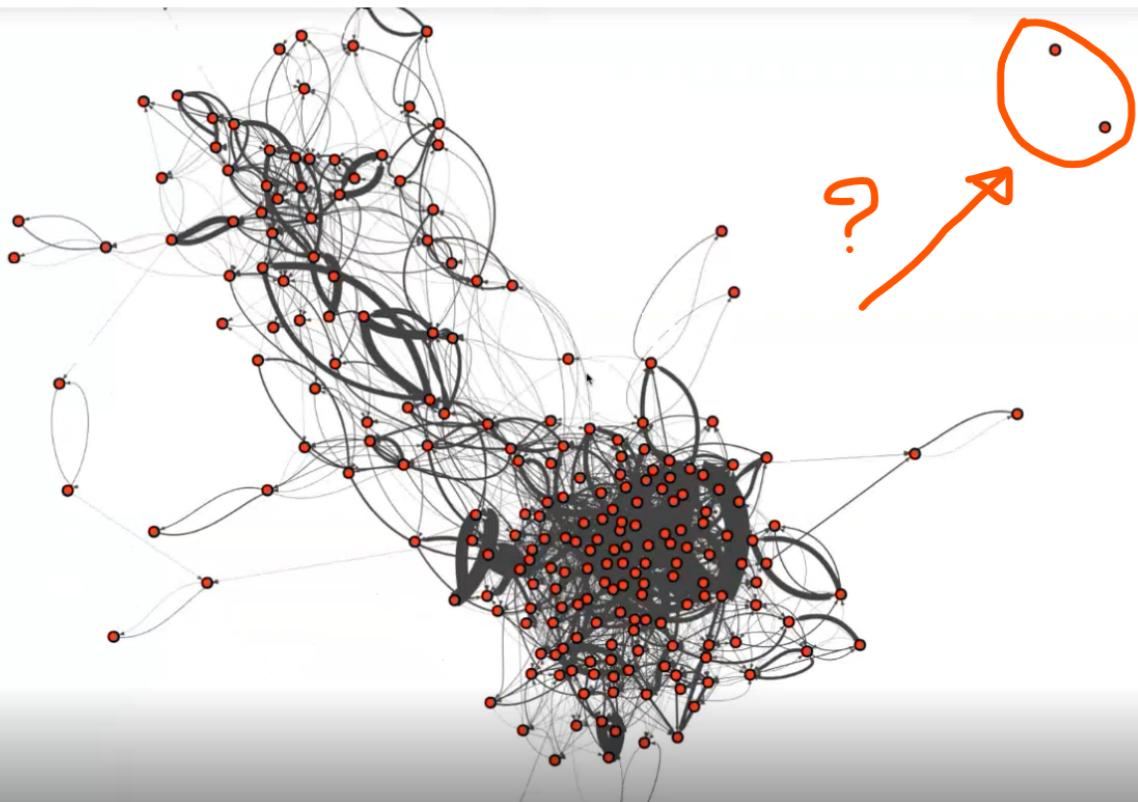
visual_style[ "bbox" ] = (600,600)
visual_style[ "label" ] = []
visual_style[ "layout" ] = fb_g_sub.layout_fruchterman_reingold()
visual_style[ "vertex_size" ] = 5
```

Con questi nodi genero un **sottografo indotto** costituito solo dai nodi dati come parametro o meglio dai link che collegano solo coppie di nodi aventi in questo esempio grado maggiore di 100.

E' un grafo indotto in quanto analizzo la connettività solo di un gruppo ristretto di nodi analizzati.

Il giant component per definizione è un grafo连通的, questo nella foto è una sottoporzione del giant component in quanto è un sottografo indotto, allora...

? come mai in questo sottografo indotto ci sono dei nodi isolati?



Perchè in questo sottografo compaiono solo nodi che hanno grado maggiore di 100 e i link collegano solo nodi con grado maggiore di 100. Nel grafo originale il nodo isolato che vediamo magari era collegato a nodi che però non avendo grado maggiore di 100 sono stati esclusi nel sottografo indotto.

### La funzione plot()

Per realizzare dei plot usiamo la funzione `plot()` che prende 2 parametri:

### Part 3. Plotting graphs

This is based on the igraph `plot()` method, which is built on the Cairo package (and specifically on the `pycairo` bindings between Cairo and Python).

The layout of the graph (i.e., the **coordinates**) of the vertices, must be computed first, with the `layout()` functions.

- many algorithms can be used to properly compute coordinates, based essentially on the edges and their weights.

```
In [11]: # Compute the layout, using one of the possible methods (Fruchterman-Reingold)
layout = toy_g.layout_fruchterman_reingold()

# Plot the graph
plot(toy_g, layout = layout)
```

Out[11]:



un oggetto grafo (`toy_g`) e un `layout` il quale deve essere un oggetto di una funzione di layout. Nella libreria igraph esistono diverse funzioni nominate `layout_*` che servono a prendere il grafo e definire delle coordinate x;y dello spazio in cui posizionare i nodi a seconda di come sono collegati tra di loro. E' possibile fare delle visualizzazioni leggermente più carine, dove invece che passare un oggetto `layout` gli si passa un oggetto che è un dizionario (`visual_style`):

```
In [12]: # Beautifying the plot with some visual style attributes
# - define a dictionary for setting the required options
# - use the dictionary as the plot parameter
# - see http://igraph.org/python/doc/tutorial/tutorial.html for the set of possible options

colors = ['red', 'blue']
shapes = ['rectangle', 'circle']
sizes = [10,30]

try:
    del visual_style
    visual_style = {}
except NameError:
    visual_style = {}

visual_style["bbox"] = (300,300)
visual_style["vertex_label"] = toy_g.vs["name"]
visual_style["vertex_label_dist"] = 2
visual_style["vertex_size"] = [sizes[toy_g_conn_comp.membership[i]] for i in range(toy_g.vcount)]
visual_style["vertex_color"] = [colors[toy_g_conn_comp.membership[i]] for i in range(toy_g.vcount)]
visual_style["vertex_shape"] = [shapes[toy_g_conn_comp.membership[i]] for i in range(toy_g.vcount)]
visual_style["layout"] = layout

plot(toy_g, **visual_style)
plot(toy_g, "toy_g_plot.pdf", **visual_style)
```

La funzione di plot() serve anche per visualizzare i componenti connessi di un grafo.

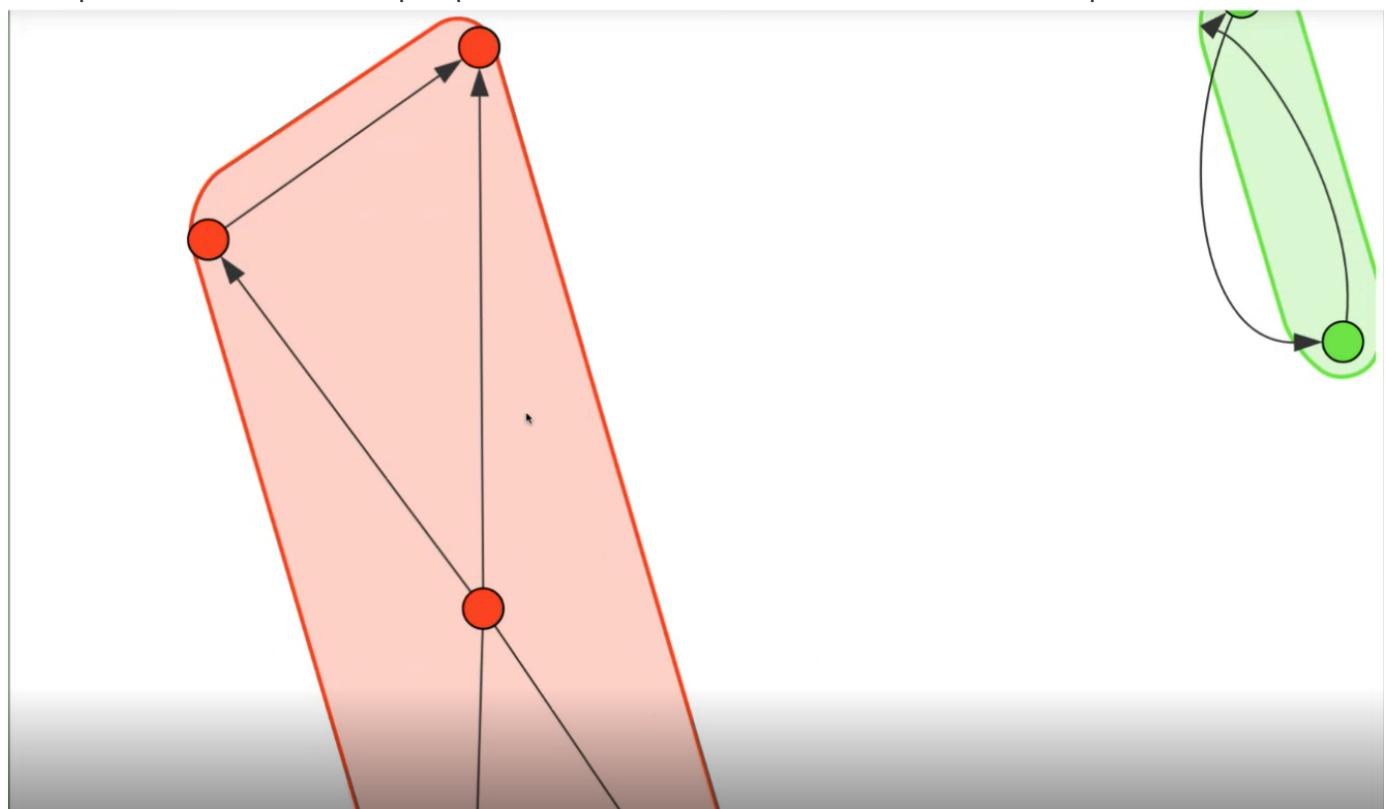
## Plotting the components

When using plot() with a VertexClustering object

- vertices belonging to different components are automatically colored differently
  - if visual style is provided, its values are used instead of default
- the mark-groups parameter can be used to highlight groups with shapes
  - True for default colors

```
In [13]: try:  
    del visual_style  
    visual_style = {}  
except NameError:  
    visual_style = {}  
plot(toy_g_conn_comp, layout=layout, mark_groups = True)
```

Se la funzione di plot la invochiamo usando come primo parametro il grafo dei componenti connessi (`(toy_g_conn_comp)`) e gli diamo un parametro layout che in questo caso è quello calcolato in qualche cella precedente, la funzione plot permette di visualizzare automaticamente i componenti connessi:



## Analisi del grado

---

```

In [14]: # degree() method
# - mode = "ALL" to consider the undirected graph
fb_deg = fb_GC.degree(mode = "all")
fb_deg[0:19]

# the maximum degree, and the ID of the node with maximum degree
max(fb_deg)
id_max = np.argmax(fb_deg)
id_max

# the set of neighbours of the node with max degree
# - NB: in case of bidirectional links, the same neighbour is counted twice if mode = 'all'
nei = fb_GC.neighbors(id_max, mode="all")
len(nei)

# the set of nodes reachable from id_max with AT MOST 1 jump
neighbours = fb_GC.neighborhood(id_max, order = 1, mode="all")
neighbours[0:19]

# the number of such nodes
# - NB: it also includes the node id_max itself (which is reachable with 0 jumps)
# - thus, the number of nodes reachable with one jump is this - 1
len(neighbours)
fb_GC.neighborhood_size(id_max, order = 1, mode="all")

```

Out[14]: [34, 2, 15, 18, 34, 41, 40, 34, 39, 90, 35, 48, 53, 5, 27, 21, 21, 17, 28, 20]

Out[14]: 314

Out[14]: 2720

Out[14]: 314

**ARELLA**

Il metodo `degree()` restituisce un array che è grande quanto il numero di vertici nel grafo e nella posizione i-esima c'è il grado del nodo i-esimo. In tal caso il parametro `mode = "all"` dunque per calcolare il grado dei nodi considero tutti i link entranti o uscenti da o in quel nodo. Se vediamo le prime 20 posizioni di questo oggetto, il fatto che il nodo di indice 0 abbia grado 34 significa che ci sono 34 link entranti/uscenti da quel nodo.

Per sapere quale nodo ha grado massimo usiamo una funzione di `numpy (np)` ovvero la `argmax`.

La funzione `neighbors()` mi restituisce il set di vicini di un particolare nodo su cui la invoco. Dunque gli passo come parametro almeno l'indice di un nodo e lui mi restituisce il set di vicini di quel nodo. Se esiste più di un link che collega gli stessi nodi vicini, questa funzione `neighbors` mi restituisce più volte lo stesso nodo perchè tale funzione parte dal nodo `id_max`, guarda tutti i link entranti o uscenti dal nodo `id_max`, vede quali sono i nodi collegati a questi link e li aggiunge ai vicini. **Dunque il numero ritornato da neighbour non è il numero di vicini di un nodo ma il numero di link incidenti o uscenti dal nodo considerato.**

Se ciò che ci interessa invece è capire quali e quanti sono i vicini di un particolare nodo, non dobbiamo usare la funzione `neighbors()` ma dobbiamo usare la funzione `neighborhood()` che ci restituisce un set di nodi che non sono però quelli raggiungibili dal nodo di interesse, sono i set di vicini **DISTINTI** raggiungibili a partire dal nodo di interesse con **AL PIU' un hop**. Dato che si tratta di AL PIU' un numero di hop pari ad order, potrebbe essere anche 0 dunque la funzione `neighborhood()` restituisce sempre anche il nodo su cui è stata chiamata come primo oggetto.

## as\_undirected()

Con questa funzione

Why is the output of `neighbourhood_size()` different from the length of `nei`?

- consider that we used a **directed** graph, and think what does it means in terms of degree and neighbours

Let's redo the same on the equivalent **undirected** graph

```
In [15]: # take the undirected version of the Giant Component
# combine_edges tells what to do with the weights (default, lost attribute; here: sum values)
fb_GC_u = fb_GC.as_undirected(combine_edges = "sum")

# Note the lower number of edges with respect to the directed version.
# This is because igraph automatically simplifies the graph (i.e., merges edges between the same
# to do so manually on a multi-edge graph: g.simplify())
# to check if the graph is simple or not: g.is_simple()
summary(fb_GC_u, verbosity = 1, edge_list_format = "edgelist", max_rows = 25)

# the maximum degree, and the ID of the node with maximum degree
fb_deg_u = fb_GC_u.degree()
max(fb_deg_u)
id_max_u = np.argmax(fb_deg_u)
id_max_u

# the set of neighbours of the node with max degree
nei_u = fb_GC_u.neighbors(id_max_u)
len(nei_u)

# the set of nodes reachable from id_max with AT MOST 1 jump
neighbours = fb_GC_u.neighborhood(id_max, order = 1, mode="all")
neighbours[0:19]

# the number of such nodes
```

stiamo creando un altro grafo partendo da Facebook trascurando la direzionalità dei link e se ci sono dei link multipli che collegano gli stessi vertici li combino e i pesi li sommo. I pesi possono essere sommati, si può prendere il maggiore, il minore, si possono fare delle svariate operazioni in base alla semantica data ai pesi e al grafo. **Adesso che ho semplificato il grafo, il numero dei vicini e il grado coincidono avendo eliminato i link multipli!**

## Degree density e CCDF

### Degree density and CCDF

One way to visualise the density of the degree is through a histogram

- discrete approximation of the density function
- bins = set of intervals for the random variable
- value in each bin = # of samples, absolute or normalised, which fall in the bin
  - i.e., the probability that the random variable takes values in the bin

igraph includes a method to compute the histogram of the degree distribution

- `degree-distribution()`

But we use Numpy/Matplotlib methods, which are more flexible

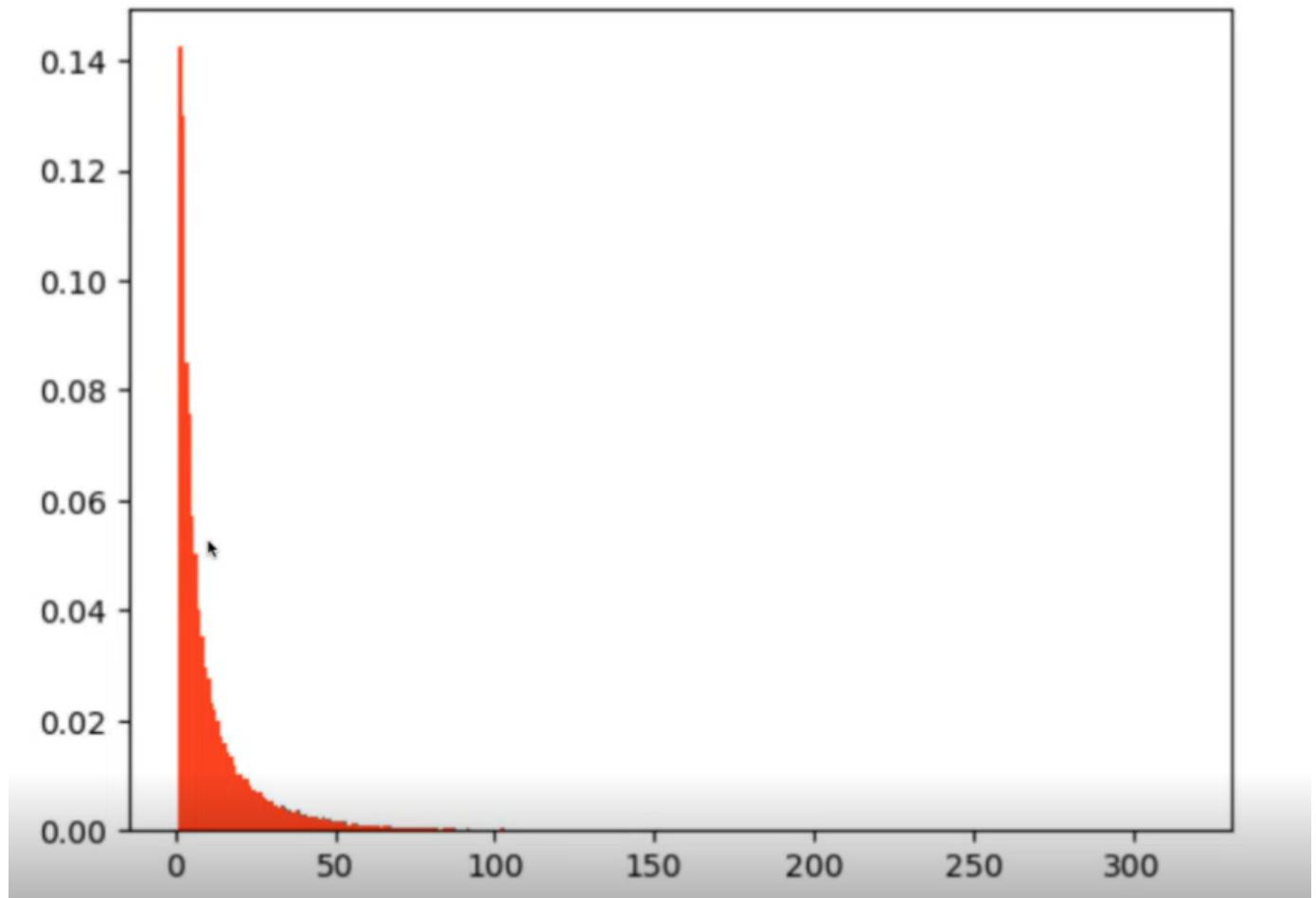
- Numpy: method `np.histogram()` computes the histogram of a set of values
- Matplotlib: method `plt.hist()` computes the histogram (via Numpy), and plots the result

```
In [16]: dd_h, dd_h_bins, _ = plt.hist(fb_deg, bins=range(1,max(fb_deg)+2), density=True, color = 'red')
```



I gradi di Facebook li abbiamo già calcolati con la Degree Analysis, dunque li abbiamo già all'interno della variabile `fb_deg`. Il primo modo per calcolare la degree distribution è usare il metodo `hist()`

che calcola un istogramma dell'oggetto che passate.



Sulle x ci sono i valori che si trovano sull'array che viene passato come parametro mentre sulle y c'è il numero di volte in cui quel valore viene trovato.

La parte interessante della degree distribution è la coda di cui noi non vediamo nulla pertanto l'istogramma non è lo strumento giusto. La cosa utile dell'istogramma non è tanto la visualizzazione quanto il fatto che in output restituisce questi oggetti:

#### Notes

- Return values
  - the values of each bin (dd-h)
  - the extremes of the bins (dd-h-bins)
    - **NB:** the number of the extremes is always the number of bins + 1!!!
  - we discard the third return value (variable -)
- Parameters
  - fb-deg is the set of values for which the histogram is computed
  - bins is a list with the extremes of the bins
    - **NB:** max(fb-deg) + 2 is because range excludes the second extreme, and the last bin represents the probability of degrees  $\geq \max(\text{fb-deg})$
  - density tells whether to normalise the values of bins to 1

```
In [17]: # how do the histogram and bins look like
dd_h[0:19]
dd_h_bins[0:19]

Out[17]: array([0.14235661, 0.13009351, 0.0851364 , 0.07569449, 0.0569927 ,
0.05014447, 0.04022479, 0.03510568, 0.02948604, 0.02762041,
0.02300184, 0.02200077, 0.02006689, 0.01722294, 0.01581235,
0.01431074, 0.01362819, 0.01196733, 0.01003344])

Out[17]: array([ 1.,  2.,  3.,  4.,  5.,  6.,  7.,  8.,  9., 10., 11., 12., 13.,
14., 15., 16., 17., 18., 19.])
```

queste percentuali sono le frequenze con cui i nodi di ciascun grado appaiono.

Abbiamo invece visto che il modo migliore di analizzare la distribuzione del grado è quella di plottarla su scale log x log.

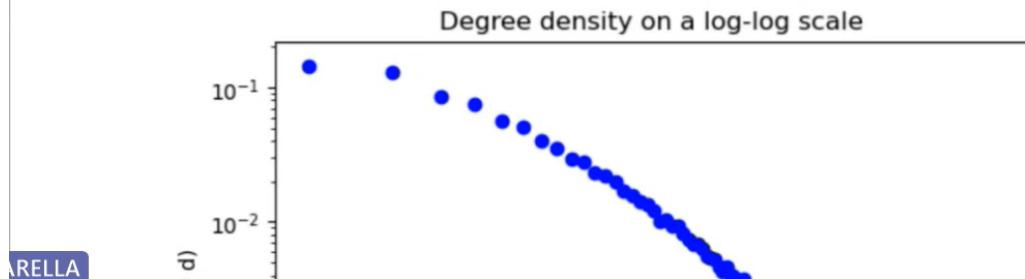
```
In [18]: # Degree density on a loglog scale
plt.loglog(dd_h_bins[:-1], dd_h, 'bo')
plt.xlabel("d")
plt.ylabel("P(Degree = d)")
plt.title("Degree density on a log-log scale")

Out[18]: []

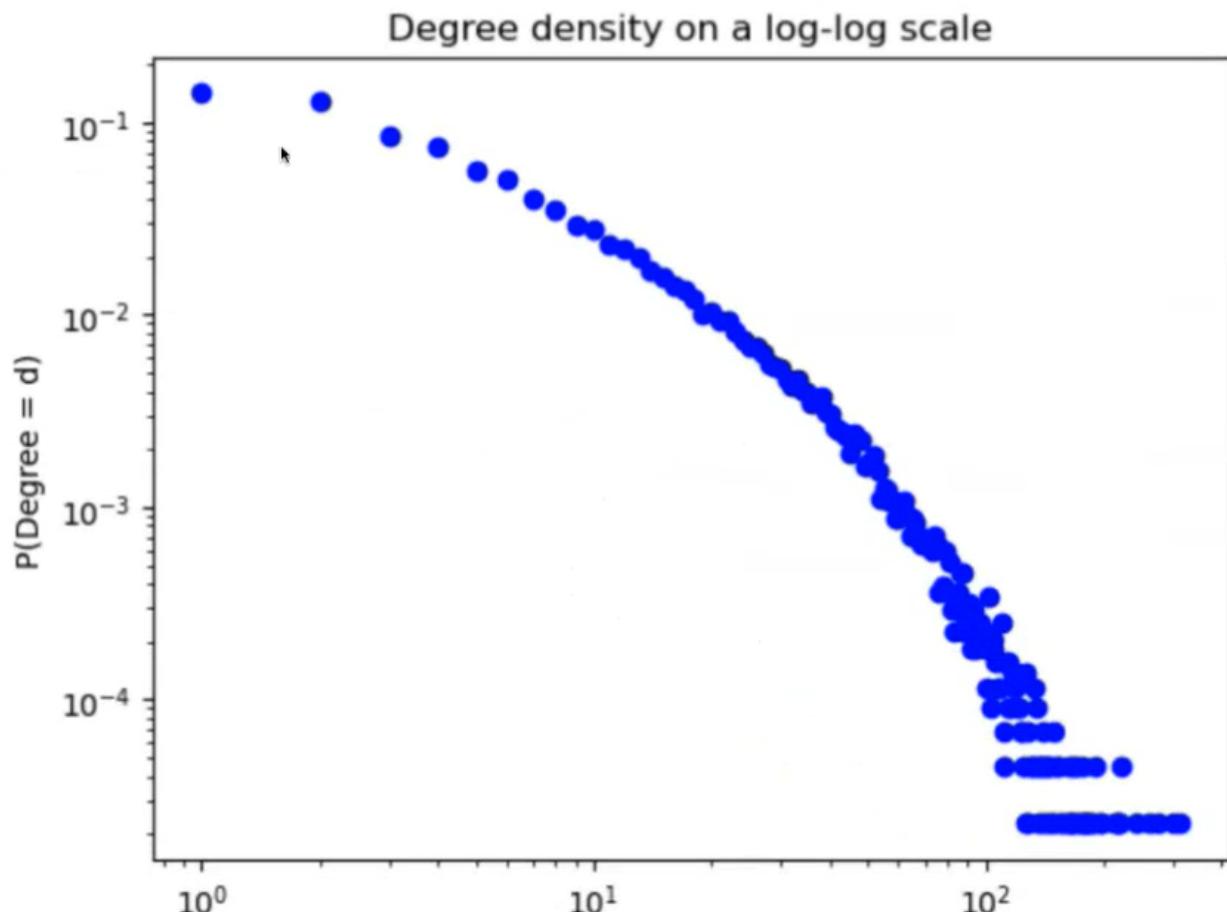
Out[18]: Text(0.5, 0, 'd')

Out[18]: Text(0, 0.5, 'P(Degree = d)')

Out[18]: Text(0.5, 1.0, 'Degree density on a log-log scale')
```



Per fare questo si usa la funzione `loglog()` che permette di rappresentare i dati in un plot log x log dove il primo parametro sono le x, il secondo parametro sono le y mentre il terzo parametro sono le istruzioni su come visualizzarlo (b = colore blue, o = visualizzati come pallini). Questa che io mi stampo su scala log x log è la densità del grado di Facebook:



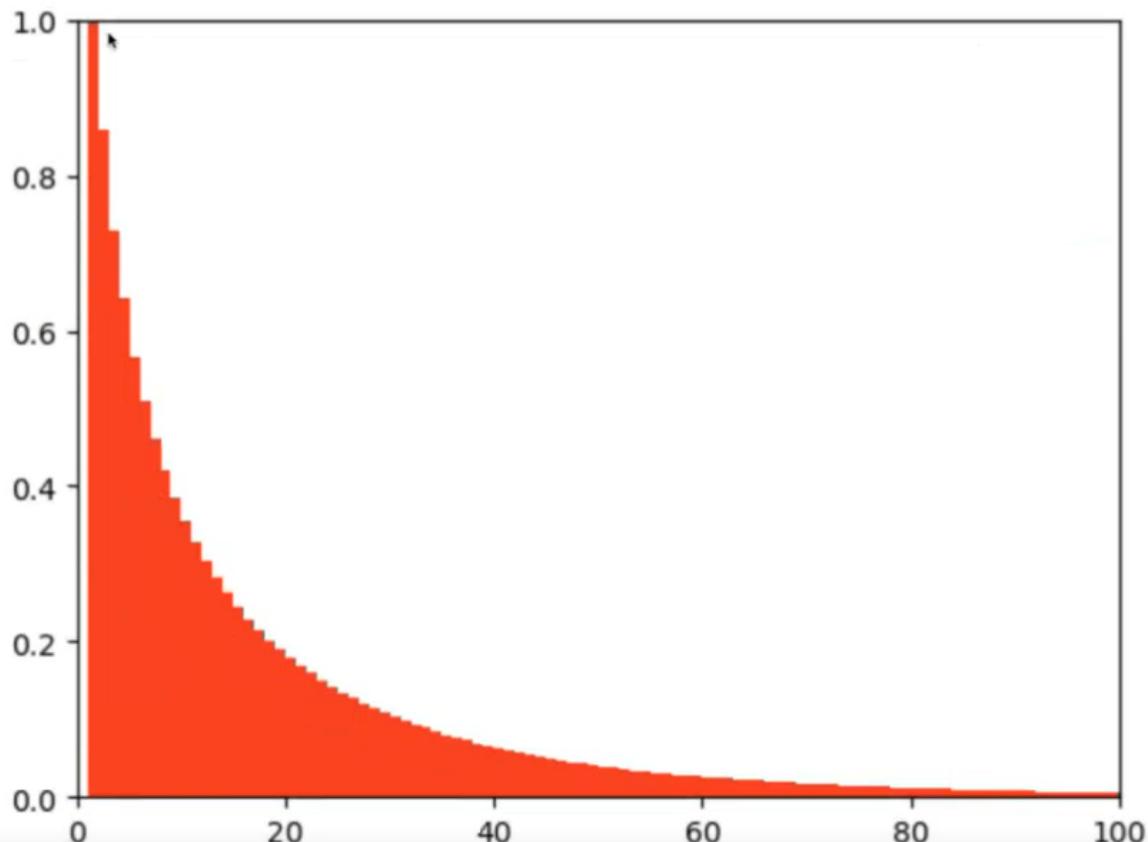
Non c'è una coda pesante ben definita con una linea retta, il motivo per cui è così è perchè sono dati reali i quali non sono mai vicini ai modelli teorici. Inoltre il dataset è abbastanza piccolo dunque l'unione di questi 2 fattori genera questo tipo di curva. Ma il fatto che l'andamento della curva sia in questo modo significa che una coda pesante è presente.

! Per analizzare però la coda del grado abbiamo detto che si usa la CCDF in quanto la densità ha questi plateau sulla coda che sono dovuti al fatto che ci sono pochi nodi con un grado alto e dunque i valori della densità empirica in questa parte della distribuzione sono molto imprecisi.

Modificando la funzione `hist()` e aggiungendo il parametro `cumulative`:

```
- we can use 2 ways
# same functions with parameter cumulative=-1 gives the CCDF
= plt.hist(fb_deg, bins=range(1,max(fb_deg)+2), density = True, color = 'red', cumulative = -1)
])
```

(0.0, 100.0, 0.0, 1.0)



Il valore della CCDF sul nodo di grado minimo deve sempre essere 1 in quanto la CCDF è la probabilità di avere nodi con grado maggiore o uguale a quello che stiamo considerando, dunque qual'è la probabilità della rete di avere nodi con grado maggiore o uguale del grado minimo? Ovviamente 1 in quanto tutti i nodi hanno un grado maggiore o uguale a quello minimo. Da questo grafico come decresce la CCDF non si riesce a dire in quanto non è su un grafico log x log. Dobbiamo usare una funzione `ECDF()` che restituisce una funzione che approssima la CDF calcolata sui gradi passati come parametro, di fatto si tratta di una funzione interpolante la CDF.

**Attenzione!** Questa è un interpolazione della Cumulative Distribution Function non della Complementary Cumulative Distribution Function. La CCDF è la probabilità che i gradi siano maggiori o uguali a, mentre la Cumulative Distribution Function è la probabilità che il grado sia minore del valore che gli diamo.

```

# 2. More general: use the ECDF function of statsmodels.distributions.empirical_distribution
# ECDF(dataset) returns a the empirical CDF computed from the dataset, which can be used as a F
# - i.e., it is possible to call ECDF(x) for any x, irrespective of the set of data from which
deg_cdf = ECDF(fb_deg)

# scale the fig size twice in length
default_sizes = plt.rcParams["figure.figsize"]
fig_sizes = (2*default_sizes[0], default_sizes[1])

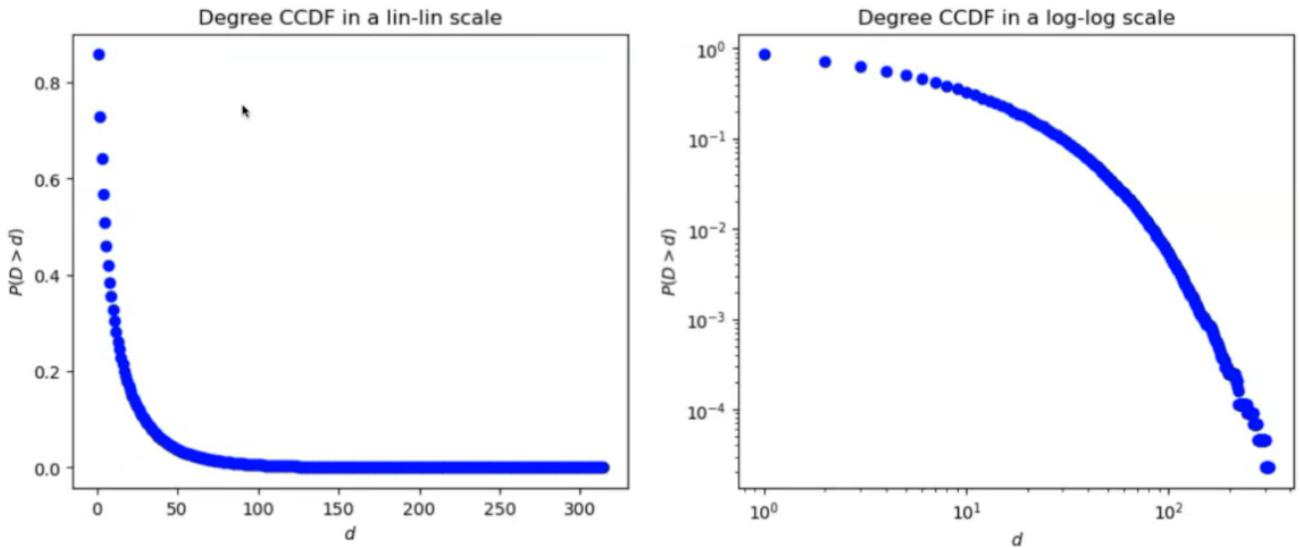
# generate a figure with 2 subplots, organised in 1 row and 2 columns
# ax1 and ax2 ("axes") are used to access the individual plots
# NB: in case of more rows, axes must be specified as list of lists, e.g., for 2x2 ((ax1,ax2), (fig, (ax1, ax2)) = plt.subplots(1, 2, figsize = fig_sizes)

# plot the CCDF in lin-lin and log-log scales
# see http://matplotlib.org/api/axes_api.html for the API of the Axis class
# see http://matplotlib.org/api/pyplot_api.html#matplotlib.pyplot.plot or the Axes.plot() docum
# for the parameters of the plot method
degs = np.arange(1,max(fb_deg)+1)

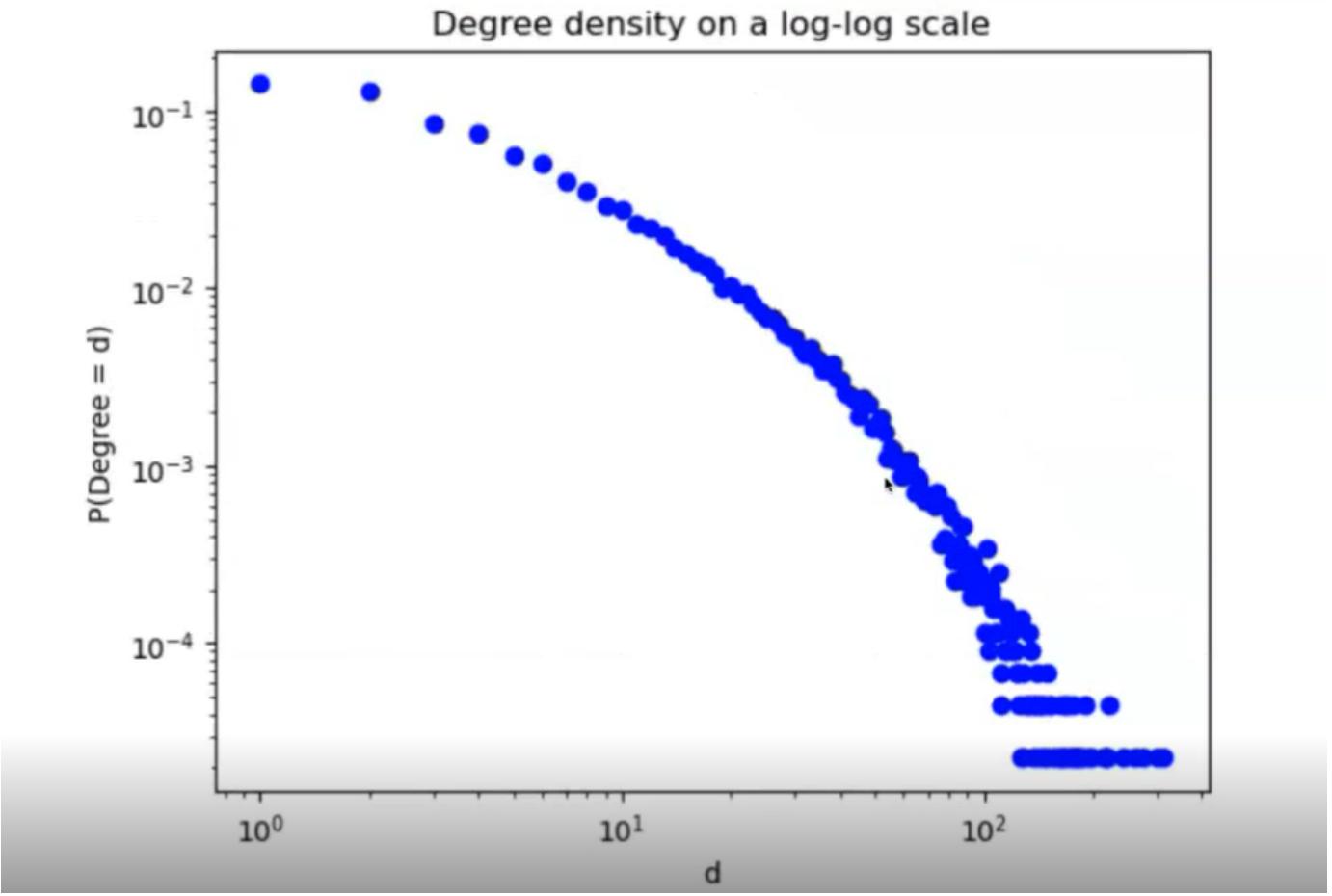
ax1.plot(degs, 1-deg_cdf(degs), 'bo')
ax1.set_xlabel("$d$")
ax1.set_ylabel("$P(D>d)$")
ax1.set_title("Degree CCDF in a lin-lin scale")

```

Per plottare la CCDF dobbiamo usare la funzione `plot()` e come x gli diamo i gradi mentre come y gli diamo `1-deg_cdf(degs)`, il fatto di fare `1-deg_cdf(degs)` ci permette di passare dalla CDF alla CCDF.



Il plot log x log della CCDF rispetto al plot precedente della densità:



è un curva molto più pulita in quanto i fenomeni dovuti al fatto che sulla parte estrema della coda ci sono pochi nodi dunque i valori della densità si stimano molto male, grazie al calcolo della CCDF si ovvia a questo problema.

## Grafi equivalenti

### Part 5. Equivalent graphs and fitting

#### Random graphs

According to the Erdős–Rényi model, a random graph is constructed by connecting nodes randomly with probability  $p$ . We build an Erdős–Rényi model equivalent to the Facebook graph

- equivalence means the same number of nodes, and the same average degree
  - thus,  $p = \frac{\langle k \rangle}{N}$

```
In [22]: # Now we compare the degree distributions for the complete fb Giant Component
er_p_GC = mean(fb_GC_u.degree()) / fb_GC_u.vcount()
er_fb_all = Graph.Erdos_Renyi(fb_GC_u.vcount(), er_p_GC)

# take only the Giant Component
er_fb = er_fb_all.connected_components(mode = "WEAK").giant()
er_fb.vcount()
fb_GC_u.vcount()

# we use GridSpecs for a finer control of the plot positioning
fig_sizes = (fig_sizes[0], 2*default_sizes[1])
f = plt.figure(figsize = fig_sizes)

# create a 2x2 Grid Specification
gs = gridspec.GridSpec(2, 2)
```

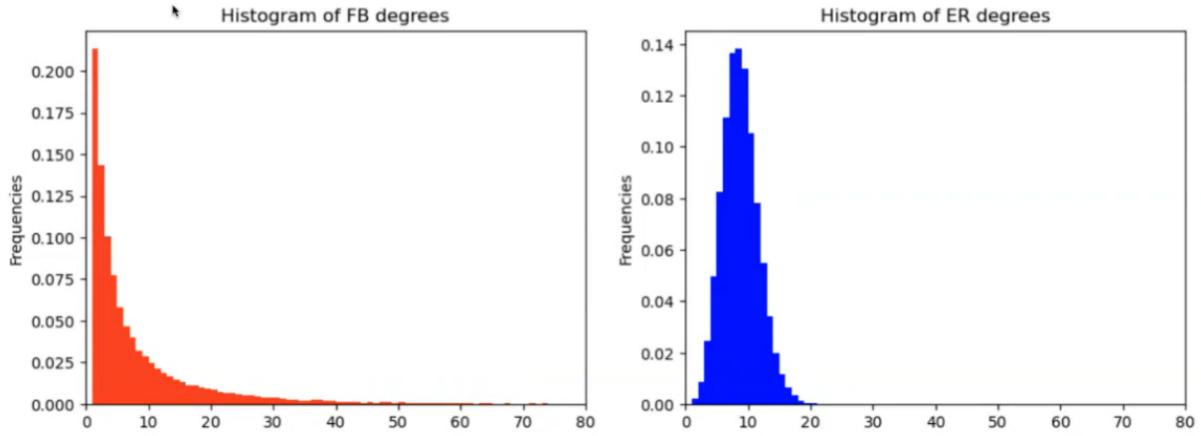
Il metodo `Graph.Erdos_Renyi()` genera un grafo prendendo 2 parametri: il numero di nodi che vogliamo nel grafo e il parametro `p` con cui i link esistono all'interno del grafo. Il parametro `p` sarà

uguale al grado medio del grafo diviso il numero di nodi. Per calcolare il grado medio devo prendermi i gradi del grafo, in questo caso stiamo usando la versione undirected del Giant Component, ma non cambierebbe se usassi anche l'altra, mi calcolo poi il valor medio dell'array dei gradi, divido per il numero dei nodi e questo è il parametro  $p$  del mio grafo Erdos-Renyi equivalente.

Questo nuovo grafo appena ottenuto, nessuno mi garantisce, anche se io sono partito da un grafo che sono sicuro essere con un unico componente连通的, che questo grafo di ER sia un unico componente连通的, dunque vado ad estrarre il giant component, mi calcolo il numero di nodi del giant component di ER e lo confronto con il numero dei nodi del mio grafo Facebook:

Out[22]: 43939

Out[22]: 43953



il grafo di ER ha 43939 nodi mentre il grafo di Facebook di partenza ne ha 43953.

La differenza in termini di nodi è molto bassa e trascurabile, quindi è vero che i grafi si confrontano a parità di nodi e di edge ma deve essere una parità cum grano salis!

Va bene considerare dunque il giant component del grafo equivalente calcolato. Dunque andiamo a plottare la CCDF anche per il grafo di ER.

Guardando le due densità già mi accorgo che sono diverse, quella di Facebook ha un andamento decrescente e sembra di vedere una coda più pesante. Se guardo il grafico di ER c'è una conformazione più a campana attorno al valore 10 che è il grado medio circa. Il terzo plot invece:

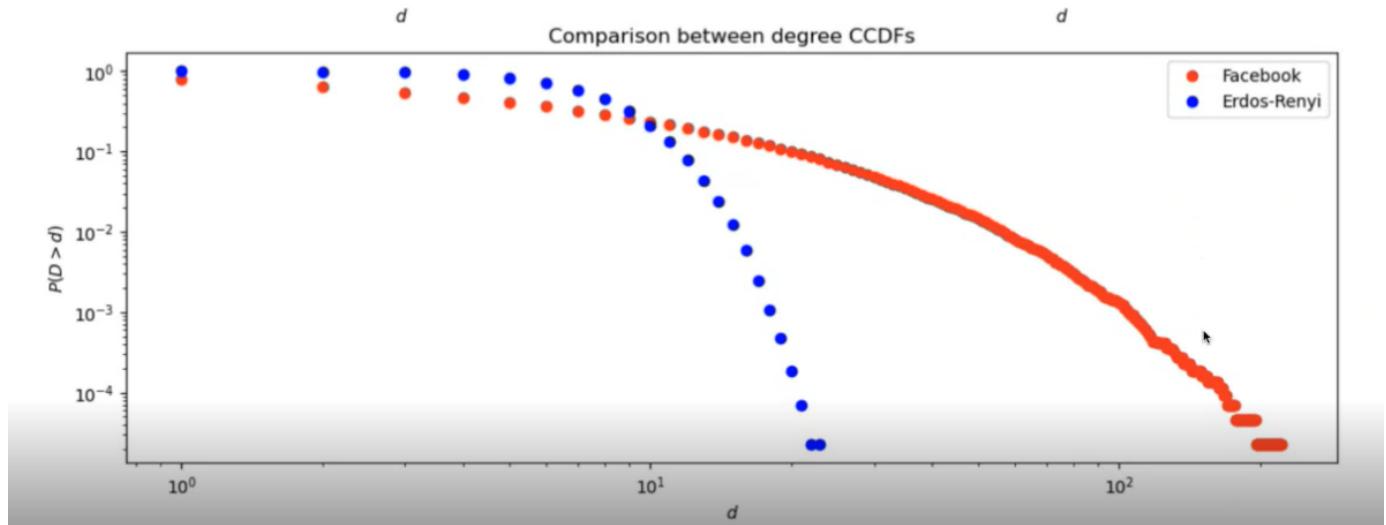
```
# compute and plot the degree CCDFs
fb_ecdf = ECDF(d_fb)
er_ecdf = ECDF(d_er)
x = np.arange(1,max(d_fb)+1)
_ = ax3.loglog(x, 1-fb_ecdf(x), 'ro', label = 'Facebook')
x = np.arange(1,max(d_er)+1)
_ = ax3.loglog(x, 1-er_ecdf(x), 'bo', label = 'Erdos-Renyi')
_ = ax3.set_xlabel("$d$")
_ = ax3.set_ylabel("$P(D>d)$")
_ = ax3.set_title("Comparison between degree CCDFs")
_ = ax3.legend(numpoints = 1)
```

ha due `loglog()` in cui plotto le CCDF di Facebook e del grafo equivalente ER.

💡 il grafo equivalente di ER non è un grafo a caso ma è quello equivalente dunque della famiglia ER più vicino possibile a quello di Facebook dunque nessun grafo è più vicino di questo al grafo di partenza.

Le 2 CCDF sono così diverse dunque non perchè è stato commesso un qualche tipo di errore in fase di configurazione, ma semplicemente non esiste un grafo di ER che mi da una distribuzione del grado più vicina a quella di Facebook ma è SOLAMENTE dovuto al fatto che il grado di Facebook è diverso

rispetto a quello di ER. In particolare la curva del grafo ER decade più velocemente del grafo di Facebook, ovvero il modello di ER è il prototipo di reti a coda leggera senza hub:



quindi il grafo di Facebook ha sicuramente una coda più pesante di un modello ER perchè la coda del grafo decade più lentamente del modello ER.

Andiamo allora a vedere se per caso la mia distribuzione può essere approssimata con una power law. Cerco quale è la distribuzione power law che meglio approssima il mio grafo:

```
# - see the plot of the CCDFs for understanding how fitting depends on xmin
xmin = 10
fit_pl = Fit(fb_GC_u.degree(), xmin = xmin)
# by computing automatically the "best" xmin value
fit_pl_auto = Fit(fb_GC_u.degree())

exp_pl_auto = fit_pl_auto.alpha
xmin_auto = fit_pl_auto.xmin
exp_pl = fit_pl.alpha
print ("PL exponents: (xmin=%d) %.2f; (auto xmin=%.2f) %.2f" % (xmin, exp_pl, xmin_auto, exp_pl))

# compute the number of nodes and edges of the FB graph to generate the equivalent static Power
N = fb_GC_u.vcount()
M = fb_GC_u.ecount()

# Equivalent graph for the fitting with fixed xmin
pl_fb_all = Graph.Static_Power_Law(N, M, exp_pl)
# the graph could not be connected, so keep the GC only
pl_fb = pl_fb_all.connected_components(mode = "WEAK").giant()

# Equivalent graph for the fitting with automatic xmin
pl_fb_auto_all = Graph.Static_Power_Law(N, M, exp_pl_auto)
pl_fb_auto = pl_fb_auto_all.connected_components(mode = "WEAK").giant()

Calculating best minimal value for power law fit
PL exponents: (xmin=10) 2.54; (auto xmin=101.00) 6.75
```

per fare questo si possono generare dei modelli di power law tramite la funzione

`Graph.Static_Power_Law()` con i seguenti parametri:

- N: il numero di nodi del grafo
- M: il numero di link
- `expl_pl`: esponente della power law che deve riprodurre con N nodi e M link

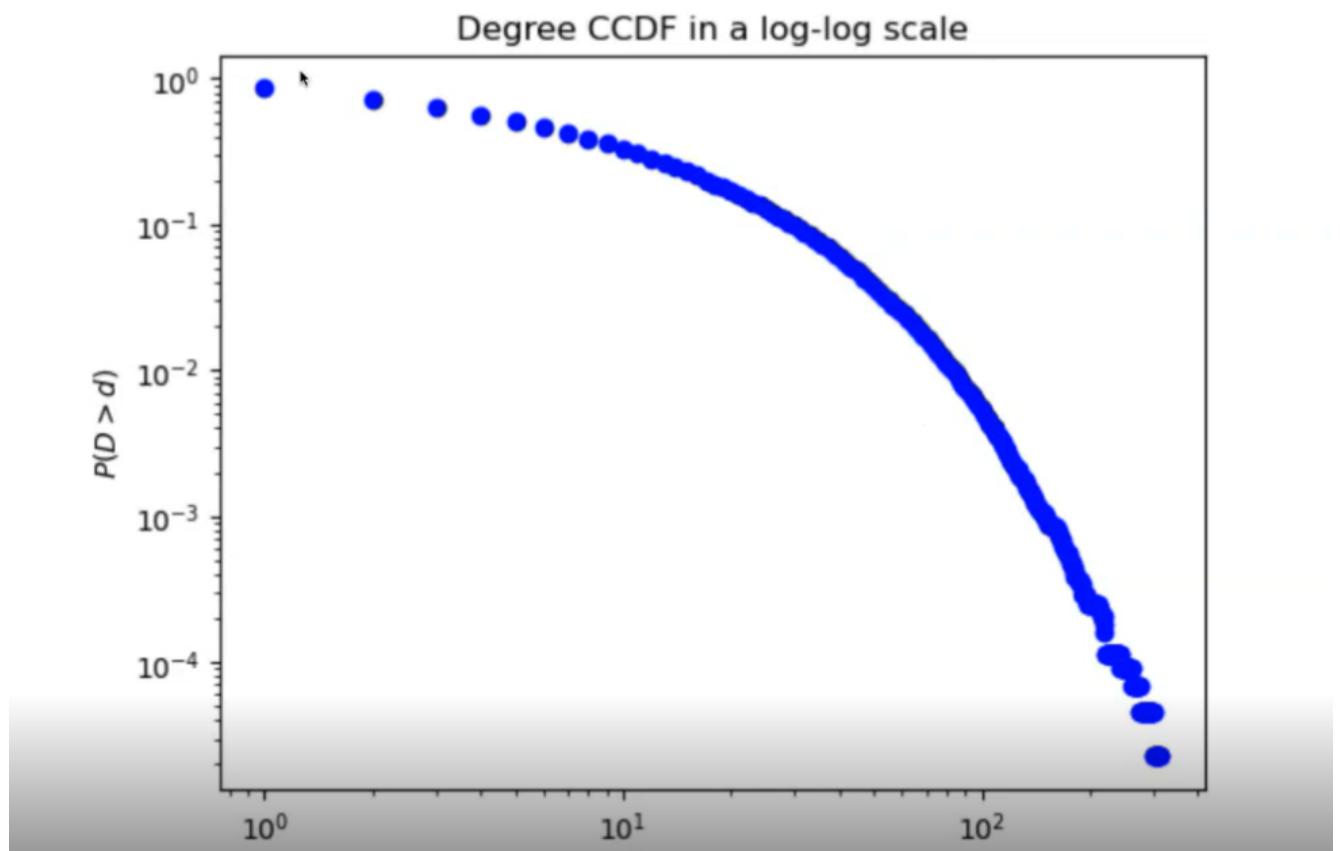
Stiamo creando un grafo che ha una distribuzione power law come vogliamo noi. Quindi non ci basta più dal grado di partenza calcolare il numero di nodi o numero dei link:

```
# compute the number of nodes and edges of the FB graph to generate the equivalent static Power
N = fb_GC_u.vcount()
M = fb_GC_u.ecount()
```

ma devo anche calcolare questo terzo parametro, che per impostarlo quale distribuzione meglio approssima la distribuzione del grado di Facebook di partenza che sarebbe il concetto di **FITTING**. Se volessi approssimare la CCDF con una power law quale sarebbe la migliore approssimazione possibile? Sto facendo il fitting e per farlo si usa la funzione `Fit()` che prende come parametro `fb_GC_u.degree()` ovvero i gradi della distribuzione da fittare. Dunque stiamo chiedendo alla funzione `Fit()` di calcolare i parametri della distribuzione power law che meglio approssimano la distribuzione dei miei gradi originali.

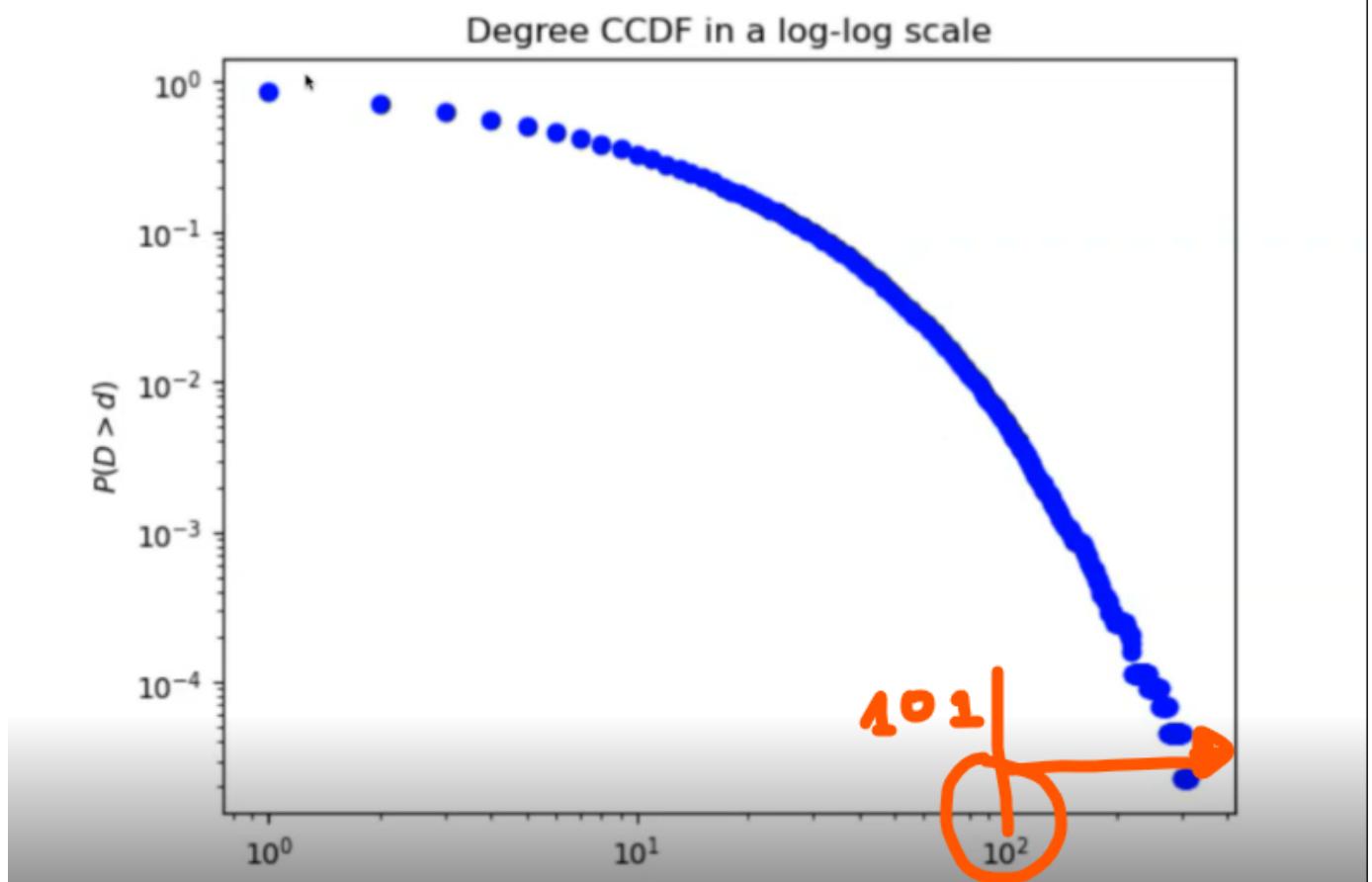
Questa funzione mi restituisce:

- un valore `.alpha` che è l'esponente della distribuzione power law (nelle slide chiamato  $\gamma$ )
- un valore `.xmin`: quando si fa un fitting con una power law il parametro `.alpha` è considerato valido per tutti i valori delle  $x$  maggiori o uguali di `.xmin`. Con la funzione di fitting noi stiamo cercando qual'è la migliore retta per approssimare questa curva:



e questo perchè il fitting power law mi restituisce la più vicina approssimazione di una power law rispetto ai miei dati e so che la power law è una retta su scala logxlog. Per valori di  $x < xmin$  non è una buona approssimazione. Se vediamo questi 2 valori abbiamo  $xmin = 101$  e  $alpha = 6.75$ . Se stessi a questi valori quello che ci ricaverei è che il miglior fitting possibile della mia powerlaw è un fitting con esponente 6.75 che vale solo per le  $x$  maggiori di 101. Questa è la migliore approssimazione da un punto di vista matematico ma da un punto di vista reale è utile? DIPENDE.

Se prendiamo la distribuzione:



101 è più o meno a  $10^2$  dunque l'output della funzione di fitting mi dice che la distribuzione dei gradi me l'approssima con un alpha circa uguale a 7 che non sarebbe una coda pesante solamente per la parte della coda che va da 100 in avanti. La nostra distribuzione ha un grado massimo che va da 100 a 300 a seconda che consideriamo i link multipli o no. Questo è abbastanza tipico dai dati reali ovvero da un punto di vista matematico, il fitting migliore di questa curva si trova su un range piuttosto limitato da 100 in avanti, e da 100 in avanti ho una coda che non è per niente pesante. Sulla coda abbiamo pochi dati quindi andarsi a fidare di così pochi dati ovvero solo della parte più estrema non è una buona idea. Qui non esiste una regola che vale sempre, ma in questo caso andare a fare un fitting solo dai dati da 100 in avanti considerando che il grado massimo è 200, 300 nella migliore delle ipotesi, non è una stima affidabile, anche perchè nei dati reali ci sono sempre degli effetti di sporcizia dovuta al fatto che ci sono pochi dati in questa parte di distribuzione. Quello che in genere si fa è imporre un valore di xmin:

```
# First, we find the best power law fit for the degree distribution
# with a fixed minimum value xmin (minimum degree for which the fitting is computed)
# - see the plot of the CCDFs for understanding how fitting depends on xmin
xmin = 10
fit_pl = Fit(fb_GC_u.degree(), xmin = xmin)
# by computing automatically the "best" xmin value
fit_pl_auto = Fit(fb_GC_u.degree())
```

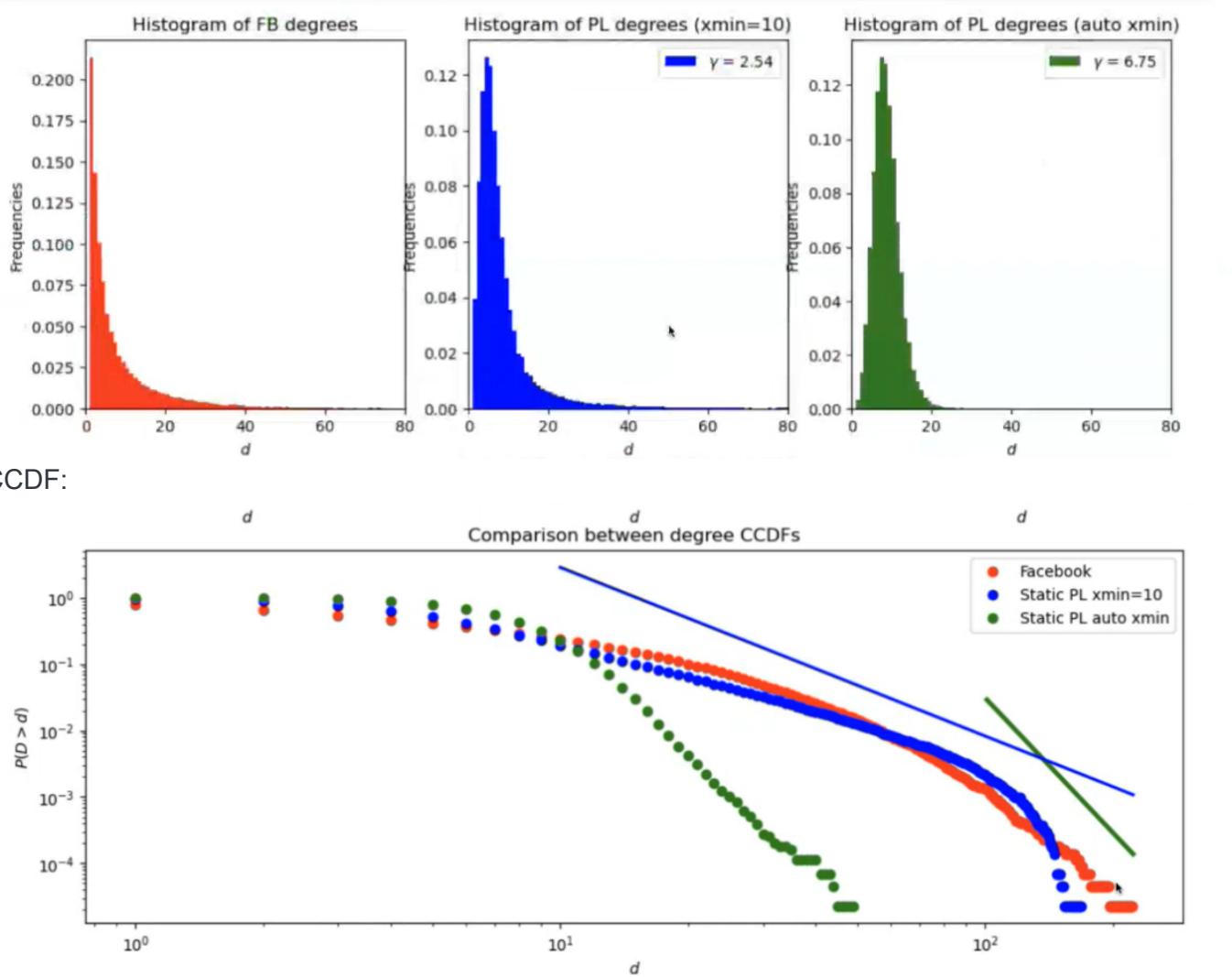
Scegliamo 10 per esperienza in quanto in genere nelle distribuzioni dei gradi della parte iniziale non ci interessa nulla di fittarla, come rule of thumb possiamo usare che il fitting va fatto almeno su un paio di decadi di dimensioni, ovvero se il grado è 300, il fitting voglio averlo almeno da 10. Sicuramente non ha senso avere un fitting fatto con pochi dati.

In questo caso mi viene tornato un valore di alpha diverso:

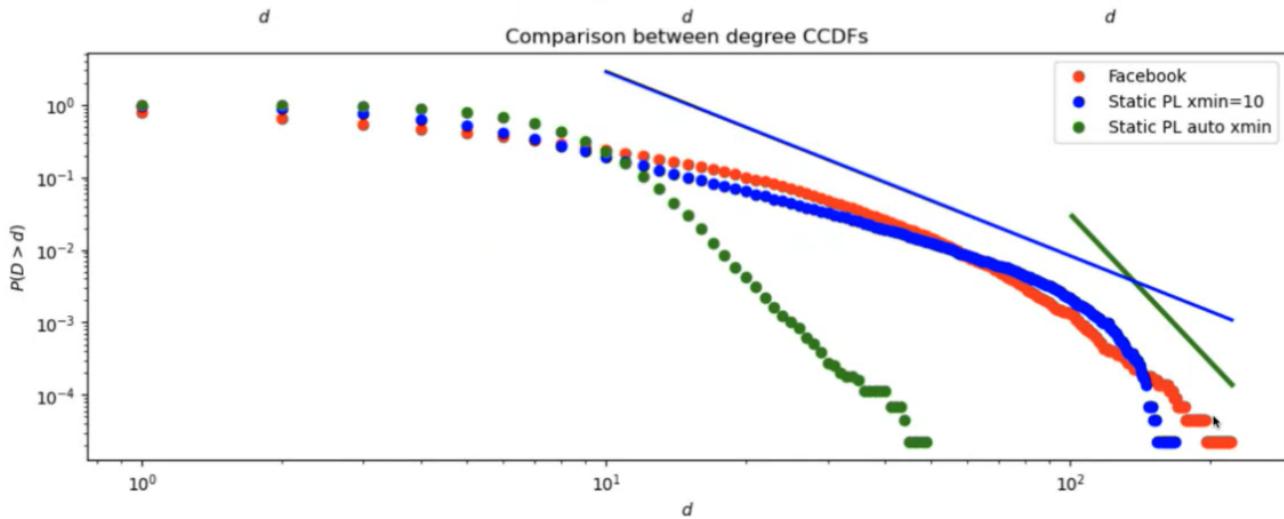
**Calculating best minimal value for power law fit**  
**PL exponents: (xmin=10) 2.54; (auto xmin=101.00) 6.75**

questo 2.54 non è meglio solo perchè rispetta la teoria di trovarsi tra 2 e 3 (poteva anche essere un altro valore lo avrei accettato lo stesso!) questo perchè è calcolato su un range di valori della distribuzione che mi copre un intervallo sufficientemente grande per essere rappresentativo. Il valore di fitting automatico è 6.75 ma è calcolato su un range di valori troppo piccolo perchè va da 101 a 300.

Tutto questo per dire che quando generiamo un grafo power law equivalente alla nostra rete dobbiamo preoccuparci non solo del numero di nodi e del numero di link ma anche della migliore approssimazione con una power law dei nostri dati. Per calcolare questa approssimazione usiamo una funzione di fitting e dobbiamo capire se lasciandolo lavorare in automatico il fitting che ci ritorna è sufficientemente esteso per essere rappresentativo. Normalmente non lo è dunque dobbiamo vedere come i gradi sono distribuiti e impostare un fitting su almeno un paio di decadi rispetto al valore massimo.



CCDF:



la rette verde ha una pendenza 6.75 come riferimento, la retta blu ha una pendenza 2.54 come riferimento.

Si vede che i pallini blu si allineano meglio ai rossi e ai verdi, questo ci conferma che il grafo equivalente calcolato con un fitting automatico in realtà non mi restituisce un grafo simile al mio, in quanto di simile tra il grafo verde e rosso c'è solo l'ultimo pezzo dei gradi estremamente elevati, da 100 in avanti. La distribuzione del grado blu approssima la distribuzione rossa in un range che va da 10 in

avanti. La parte estrema della coda è molto sporca da un punto di vista dei dati perchè ci sono pochissimi dati quindi tutta l'area da 100 in avanti non è affidabile ne nei dati ne nella approssimazione.

La rete Facebook è dunque approssimata male da una rete ER, dunque sicuramente non ha una coda leggera, è approssimata abbastanza bene da una distribuzione a coda pesante e l'approssimazione che abbiamo calcolato su un range di valori di circa 2 decadi rispetto al valore massimo ha un  $\gamma$  di 2.54 di tipo power law.

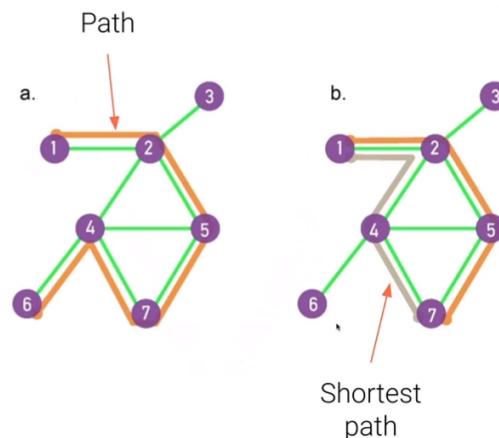
23/06/2023

## Roadmap



## Path, shortest path, minimum cost path

- Path between  $i$  and  $j$ 
  - any sequence of intermediate nodes connecting  $i$  and  $j$
- Shortest path (between  $i$  and  $j$ ),  $d_{ij}$ 
  - Path with the lowest number of intermediate hops
  - There can be more than one
- Minimum cost path (between  $i$  and  $j$ )
  - For weighted network, where weights are a measure of cost
  - Path with the overall minimum cost
  - Can be topologically longer than the shortest path!



Il path tra due nodi è qualunque sequenza di nodi intermedi che permettono di raggiungere il secondo nodo partendo dal primo. Fra tutti i path possibili ci interessano in particolare gli **shortest path** ovvero i cammini minimi all'interno della rete ovvero con il numero minimo di hop.

Quando ad un link associamo un peso, spesso è un concetto legato al costo come ad esempio il costo

del routing di internet. Quando si parla dunque di minimum cost path si parla del path che ha il costo minimo considerando la somma dei costi lungo tutto il path. **Non è detto che lo shortest path coincide però con il path a costo minimo!**

## Average path length

---

- Average value of all shortest paths in the network  $d = \frac{1}{N(N-1)} \sum_{i,j=1,N; i \neq j} d_{i,j}$
- Can be computed only on a connected component
  - As  $d_{ij} = \infty$  for disconnected pairs of nodes
- Typically, computed on the Giant Component

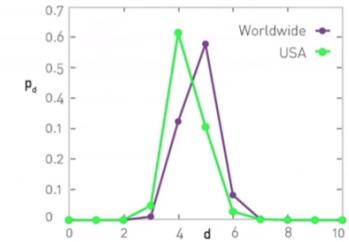
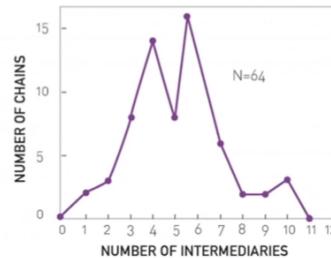
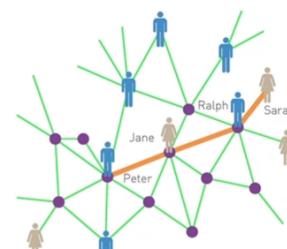
Ci interessa una metrica complessiva su tutta la rete ovvero il **valor medio degli shortest paths**. Si tratta di una media aritmetica dove viene calcolata per tutte le coppie di nodi  $(i, j)$  la lunghezza dello shortest path topologico tra i nodi e se ne fa la media aritmetica dividendo per  $N(N-1)$  in quanto in una rete di  $N$  nodi  $N(N-1)$  è il numero complessivo di coppie di nodi possibili.

Questo valore può essere calcolato solo su un connected component. Se voglio calcolare la lunghezza tra due nodi di 2 componenti diversi mi verrebbe lunghezza del path infinita quindi avrebbe poco senso.

## Small Worlds

---

- Six degrees of separation
  - Everyone is linked with anyone else by 6 intermediate steps
- Empirical evidence
  - Milgram experiment (1967)
  - Facebook (2011)
- Why?



Questo indice è stato uno dei primi ad essere analizzato e che diede luogo ai famosi "6 gradi di separazione delle società umane", ovvero ciascuno di noi è connesso a qualcun altro nel mondo tramite un cammino non più lungo di 6 "hop" (mediamente). Le prime evidenze di questo fenomeno

vengono da un famoso esperimento di Milgram, un sociologo americano, che alla fine degli anni 70 fece una serie di esperimenti:

ha preso una serie di lettere dandole a un certo numero di persone degli USA, indirizzate ad altre persone di altre città degli USA. Il compito era quello di consegnare la lettera a mano ad una persona che si reputava fosse in grado di avvicinarla il più possibile alla destinazione tramite alcune sue conoscenze di altre persone. Le lettere erano usate come traccianti per tracciare le connessioni sociali, che venivano segnate sul retro della busta, ogni volta che la lettera veniva consegnata ad un'altra persona. Moltissime di queste lettere non sono state consegnate ma di quelle consegnate (circa 70) avevano tutte mediamente un numero di hop intermedi pari a 6.

Questo esperimento è stato replicato in diversi modi e casi, più recentemente quando Facebook e Twitter erano diventate piattaforme di fama mondiale, si è cercato di capire se questa ipotesi era confermata ed effettivamente si può vedere come mediamente il grado di separazione sia pari a 5 circa.

Questa proprietà si trova su tutte le reti complessi sia sociali che tecnologiche. Si può spiegare analiticamente con i modelli di grafi equivalenti.

Prendendo il modello di ER:

## Average Path Length

---

- ❑ For Random Graphs (ER model)  $\langle d \rangle \approx \frac{\ln N}{\ln \langle k \rangle}$

- ❑ Meaning, **very short**, as  $\ln N \ll N$

- ❑ Does it hold in **real** networks?

- ❑ Basically, **yes**
  - ❑ But in some cases the real path length is even **shorter**

Network	N	L	$\langle k \rangle$	$\langle d \rangle$	$d_{max}$	$\ln N / \ln \langle k \rangle$
Internet	192,244	609,066	6.34	6.98	26	6.58
WWW	325,729	1,497,134	4.60	11.27	93	8.31
Power Grid	4,941	6,594	2.67	18.99	46	8.66
Mobile-Phone Calls	36,595	91,826	2.51	11.72	39	11.42
Email	57,194	103,731	1.81	5.88	18	18.4
Science Collaboration	23,133	93,437	8.08	5.35	15	4.81
Actor Network	702,388	29,397,908	83.71	3.91	14	3.04
Citation Network	449,673	4,707,958	10.43	11.21	42	5.55
E. Coli Metabolism	1,039	5,802	5.58	2.98	8	4.04
Protein Interactions	2,018	2,930	2.90	5.61	14	7.14

si trova che la lunghezza dello shortest path è uguale alla formula presentata in immagine, ovvero il rapporto tra il logaritmo del numero dei nodi e il logaritmo del grado medio. Questa formula ci dice che ci possiamo aspettare dei path corti in quanto se la rete è fatta da un 1 miliardo di nodi abbiamo il logaritmo che è 3. Questa proprietà si ritrova in modo pervasivo in tutte le reti che analizzeremo. In realtà lo shortest path può essere anche più corto perché dipende dalla distribuzione del grado.

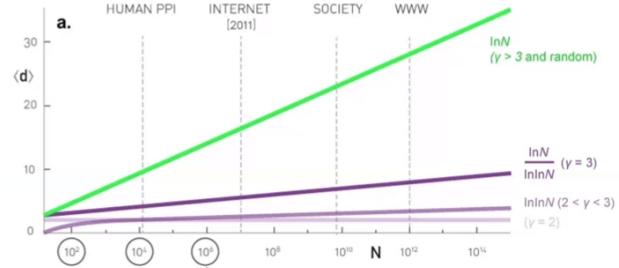
## Average Path Length

- In case of power laws, the average path length becomes much shorter

$$\langle d \rangle \sim \begin{cases} \text{const.} & \gamma = 2 \\ \frac{\ln N}{\ln \ln N} & 2 < \gamma < 3 \\ \frac{\ln N}{\ln N} & \gamma = 3 \\ \frac{\ln \ln N}{\ln N} & \gamma > 3 \end{cases}$$

- Effect of hubs

- The bigger the hubs, the more likely anyone will "bump" into them
- Hubs working as "shortcuts" connecting different parts of the network



La distribuzione del grado determina l'andamento dello shortest path. Per reti a coda pesante, di tipo power law, gli shortest path sono ancora più corti del logaritmo di  $N$ .

- Quando  $\gamma > 3$  benchè la distribuzione del grado sia simile a una power law, si tratta di una power law "poco power" dunque il comportamento è più simile a una coda leggera dunque più simile a un modello di ER.
- Quando  $2 < \gamma < 3$  la lunghezza degli shortest path si riduce ulteriormente in quanto si passa al  $\ln \ln N$ .
- Quando  $\gamma = 2$ , ovvero reti a coda pesante, la lunghezza degli shortest path non cambia è costante.

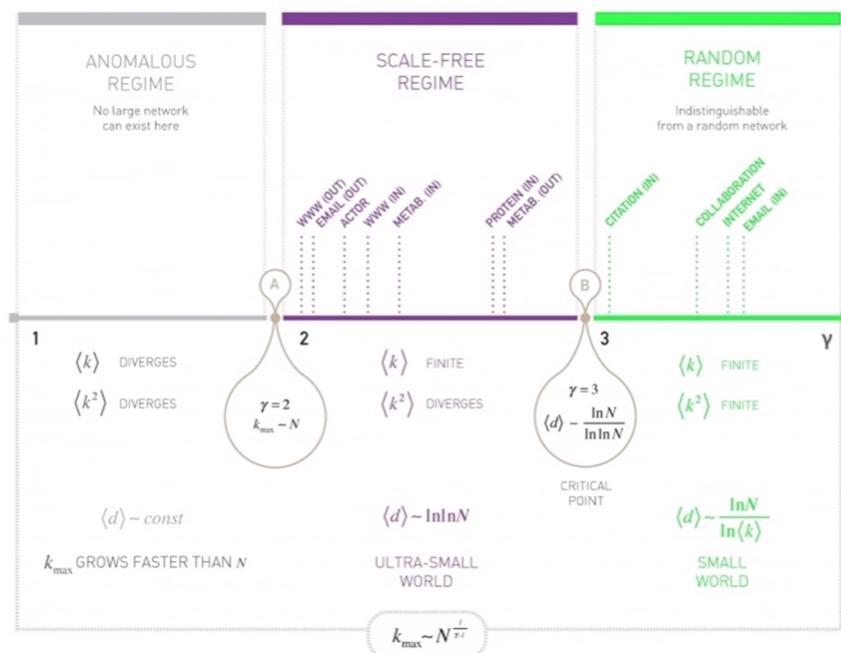
❓ Perchè più la coda è pesante e più lo shortest path è piccolo?

Più la coda è pesante più gli hub sono grandi, dunque partendo da un nodo generico o questo è connesso direttamente con un hub oppure è a pochi hop dall'hub.

Gli hub in una rete a coda pesante lavorano come tunnel, permettono di navigare da una parte all'altra della rete in pochi hop. Più questi hub sono grandissimi come nel caso di power law con  $\gamma = 2$  e più la lunghezza degli shortest path rimangono costanti in quanto gli hub attraggono una marea di link.

Tante proprietà della rete sono dunque comprensibili a partire dalla distribuzione del grado:

## At-a-glance summary



Questa proprietà dei path quando si analizza una rete in genere si usa come *negative check*. Questa proprietà è utile quando analizzando i dati ci sono delle lunghezze di path molto più lunghe di quelle aspettate stimandole con la formula vista. Questo succede quando la rete ha al suo interno delle strutture che fanno sì che le lunghezze dei cammini minimi siamo molto più grandi di quello che uno si aspetta dalla normalità delle reti. Queste strutture sono a catena ovvero strutture dove ci sono dei set di nodi collegati a destra e sinistra con un nodo e con nessun altro.

Quando ci sono lunghe catene di questo tipo è il tipico caso in cui troviamo lunghezze di cammini minimi significativamente più lunghe del  $\ln N / \ln k$ .

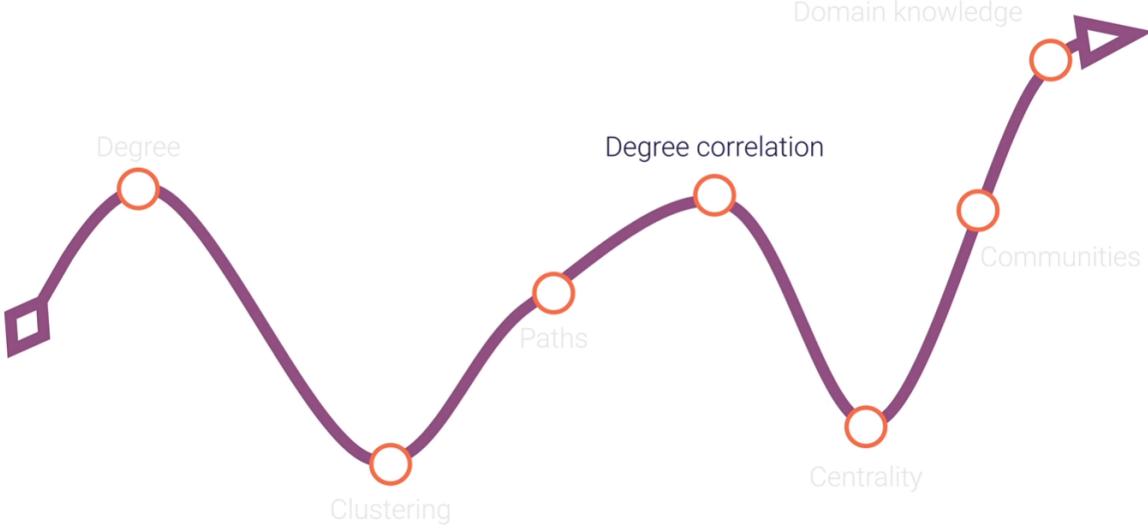
In questi casi vuol dire che ci sono comportamenti strani nel sistema e vanno analizzate maggiormente. Qualche anno fa il gruppo di ricerca del professore durante l'analisi di alcune transazioni blockchain si resero conto che veniva uno shortest path più lungo di quello atteso. Hanno trovato che nella catena delle transazioni analizzate si stavano verificando delle catene di microtransazioni che servivano solamente ad aumentare la reputazione di alcuni particolari nodi che facevano da miner.

Nelle reti di 50000 nodi se ci troviamo una lunghezza dei path di 20 anziché 4,5, o 6 allora è un campanello di allarme.

## Degree correlation

## Roadmap

---



La correlazione dei gradi ci indica se i nodi speciali (hub) sono collegati con altri hub o no.

## Assortativity and Disassortativity

---

- Do hubs connect to other hubs?
- Or do hubs connect to “normal” nodes?
- Formally
  - “Hubs connect to hubs”: **assortativity**
  - “Hubs connect to normal”: **disassortativity**
  - “Hubs connect to anyone”: **neutral**



Quando gli hub sono connessi con altri hub e i nodi normali sono connessi con altri nodi normali parliamo di reti **ASSORTATIVE**.

Quando gli hub sono connessi con nodi normali parliamo di **DISASSORTATIVITA'**.

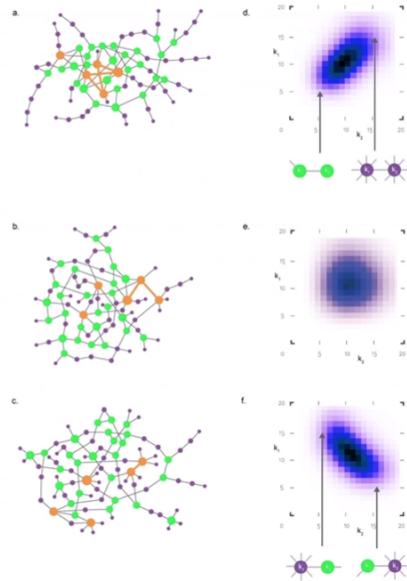
Quando non c'è un trend specifico parliamo di reti **NEUTRALI**.

Un modo per visualizzare le proprietà in termini di assortitività è la seguente:

## Assortativity and Disassortativity

- Exactly same degree distribution

- a. and d.: assortative
- b. and e.: neutral
- c. and f.: disassortative

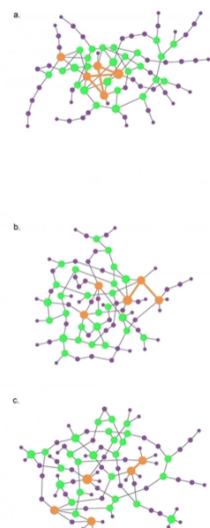


prendendo la rete e tutti i link esistenti al suo interno, in un grafico 2D sulle ascisse mettiamo, preso un link, il grado del nodo a destra del link mentre sulle ordinate ci metto il grado del nodo a sinistra del link. Se la rete è assortativa come nel primo caso i punti mi si allineano come nel caso in alto a destra nel pannello **d**.

Una rete disassortativa è il pannello **c** mentre una rete neutrale è la rete nel pannello **b**.

## Assortativity and Disassortativity - why do we care?

- In **assortative** networks hubs form a strong cluster
  - Thus, the network is much more **cohesive**
- Is it good or bad? **Depends**
  - **Resiliency:** assortative networks are more robust
    - If you kill one hub, the other will “replace” its role
  - **Epidemics:** assortative networks are more vulnerable
    - If epidemic reaches a hub, it will reach very easily all the others
    - If you “quarantine” a hub, the epidemic will spread through the others



L'assortatività mi determina se gli hub sono connessi tra di loro. Questo ha una serie di proprietà importanti come per esempio se la rete è robusta, in quanto se distruggo un hub in una rete assortativa dove gli hub sono connessi tra di loro, gli altri hub esistenti rimpiazzano l'hub appena distrutto dunque la rete resiste abbastanza bene mentre in una rete disassortativa gli faccio molto più male alla rete in quanto disconnetto in un colpo solo tutto un set di nodi in quanto gli hub sono connessi a nodi a grado basso.

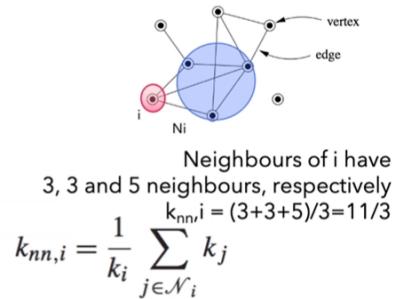
Si usano indici più rigorosi per calcolare l'assortatività:

## **k<sub>nn</sub>**

### Measuring assortativity - k<sub>nn</sub> (k) index

- ❑ Idea: measure the **expected degree of neighbours**, for a generic node with degreee  $k$
- ❑ Algorithm

- ❑ Phase a: compute  $k_{nn,i}$ 
  - ❑ Take a generic node  $i$ , whose degree is  $k_i$
  - ❑ Take all neighbours of node  $i$ , they will be  $N_i$
  - ❑ Compute the average degree of the neighbours of  $i$
  - ❑  $k_{nn,i}$  is thus the expected degree of a neighbours of node  $i$
- ❑ Phase b: compute  $k_{nn}(k)$ 
  - ❑ Group together all nodes with the same degree,  $G_k$
  - ❑ Compute the average value of  $k_{nn,i}$  for each group
  - ❑ This is exactly  $k_{nn}(k)$ : the expected degree of neighbours of nodes with degree  $k$



$$k_{nn}(k) = \frac{1}{N_k} \sum_{i \in G_k} k_{nn,i}$$

L'assortativa della rete si misura con una funzione  $k_{nn}(k)$  dove  $k$  sta per il grado della rete ovvero è una proprietà che hanno tutti i nodi di un certo grado. Si calcola in 2 modi. Se prendiamo come esempio la rete semplice mostrata nella slide, per ciascun nodo della rete, si calcola l'indice  $k_{nn,i}$ , ovvero relativo al nodo  $i$ . Si calcola prendendo i vicini del nodo  $i$  che sono in questo caso i 3 nodi nella palla blu, mi calcolo il loro grado e mi calcolo il valor medio.

Dunque per il nodo  $i$ -esimo evidenziato in rosso, l'indice  $k_{nn,i}$  è uguale a  $(3+3+5)/3$  ovvero circa 4.

❓ Fisicamente che significato ha questo indice  $k_{nn,i}$ ? Si tratta del valor medio dei vicini.

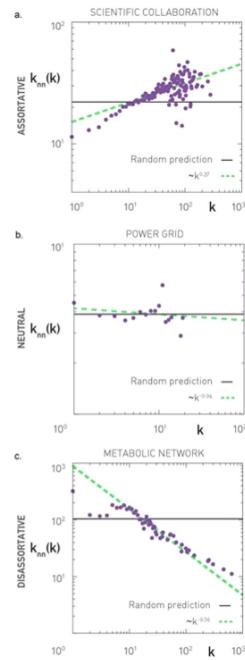
Plotando tutti i  $k_{nn,i}$  si ottiene un grafico poco chiaro come vedremo. Possiamo allora raggruppare tutti i nodi  $i$  con uguale grado e mi calcolo il valor medio dei loro valori  $k_{nn,i}$ .

Fisicamente questo valore  $k_{nn}(k)$  è riferito al grado  $k$ -esimo è il valor medio del grado  $k_{nn,i}$  per tutti i nodi che hanno grado  $k$ , ovvero vuol dire che *se prendo un generico nodo della rete con grado  $k$ , chiudo gli occhi e prendo un suo vicino, mi aspetto di aver preso un nodo con grado  $k_{nn}(k)$* .

A questo punto possiamo plottare questo indice:

## Analysis of $k_{nn}(k)$

- Assortative networks:  $k_{nn}(k)$  increases
- Disassortative networks:  $k_{nn}(k)$  decreases
- Neutral networks:  $k_{nn}(k)$  constant, or a blob of points



Se il valore di  $k_{nn}(k)$  è crescente come nel plot in alto, vuol dire che quando considero il grado medio dei nodi, vuol dire che il grado medio dei vicini è alto ovvero più cresce il grado dei nodi più cresce anche quello dei vicini dunque si tratta di una rete assortativa.

Se invece ho un andamento descrescente vuol dire che più cresce il grado e più il grado dei vicini sarà basso ovvero una rete dissartotativa.

Invece se ho un allineamento dei punti su una retta orizzontale vuol dire che non ho alcun andamento particolare.

Questo indice è un indice dettagliato (si tratta di una funzione più che un indice) che si può calcolare per tutti i gradi della rete.

**Esiste poi un altro indice che invece mi restituisce un valore solo per tutta la rete.**

## Coefficiente di correlazione di Pearson

## Measuring assortativity - degree correlation

### □ Pearson's Correlation coefficient

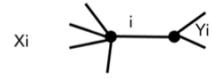
- Take a particular edge  $i$ , and consider the degrees at the endpoints

- $x_i = \# \text{edges on one end of } i$

- $y_i = \# \text{edges on the other end of } i$

- Compile 2 vectors  $X$  and  $Y$ , with all  $x_i$  and  $y_i$  values

- Compute the standard Pearson's correlation between  $X$  and  $Y$   $r = \frac{\sum_{i=1}^n (X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum_{i=1}^n (X_i - \bar{X})^2} \sqrt{\sum_{i=1}^n (Y_i - \bar{Y})^2}}$



### □ Physical meaning

- $r$  positive (positive correlation): low(high) degree nodes are preferentially linked with other low(high) degree nodes → assortativity
- $r$  negative (negative correlation): low/high degree nodes are preferentially linked with high (low) degree nodes → disassortativity
- $r \sim 0$ : neutral

Prendiamo tutti i link della rete, dunque consideriamo il link  $i$ -esimo, un nodo lo battezzo come nodo di destra e uno come di sinistra, non importa quale metto a sinistra o destra. Dunque  $x_i$  è il grado del nodo di sinistra relativo al link  $i$ -esimo.

Mi prendo tutti gli  $x_i$  e secondo l'ordine di processamento usato per scorrere gli edge li metto in un array  $X$ . Stesso discorso per il vettore  $Y$ . Lo stesso nodo contribuisce più di una volta alla stringa  $X$  o  $Y$ . Ad esempio il nodo che ha  $x_i$  contribuirà 5 volte in quanto quel nodo ha grado 5.

Questi 2 vettori avranno la stessa dimensione, e nella posizione  $i$ -esima avrò il grado dell'altro nodo che quel link  $i$ -esimo collega. Le posizioni su questi 2 vettori sono fondamentali in quanto mettono in relazione 2 nodi che sono collegati da un link nella rete. A questo punto si usa l'indice di correlazione tra il vettore  $X$  e il vettore  $Y$  che è un valore tra -1 e 1. Se l'indice  $r$  è positivo allora tutte le volte che nel vettore di sinistra delle  $X$  ci trovo un valore alto anche nella corrispondente posizione nel vettore di destra ci trovo un valore alto dunque la rete è assortativa, viceversa se  $r$  è negativo significa che ho una rete disassortativa. Se  $r$  è vicina allo 0 la rete è neutrale.

**Su reti reali il suo range di valori è tra -0.4 e 0.4.**

## Assortativity in real networks

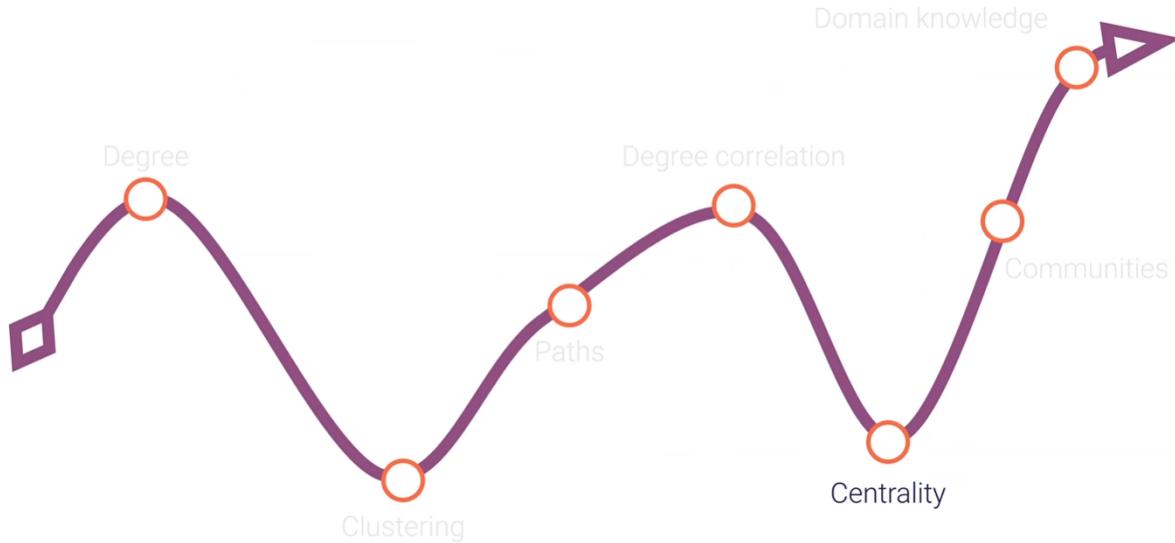
	network	type	$n$	$m$	$z$	$\ell$	$\alpha$	$C^{(1)}$	$C^{(2)}$	$r$	Ref(s.).
social	film actors	undirected	449 913	25 516 482	113.43	3.48	2.3	0.20	0.78	0.208	20, 416
	company directors	undirected	7 673	55 392	14.44	4.60	—	0.59	0.88	0.276	105, 323
	math coauthorship	undirected	253 339	496 489	3.92	7.57	—	0.15	0.34	0.120	107, 182
	physics coauthorship	undirected	52 909	245 300	9.27	6.19	—	0.45	0.56	0.363	311, 313
	biology coauthorship	undirected	1 520 251	11 803 064	15.53	4.92	—	0.088	0.60	0.127	311, 313
	telephone call graph	undirected	47 000 000	80 000 000	3.16	—	2.1	—	—	—	8, 9
	email messages	directed	59 912	86 300	1.44	4.95	1.5/2.0	—	0.16	—	136
	email address books	directed	16 881	57 029	3.38	5.22	—	0.17	0.13	0.092	321
	student relationships	undirected	573	477	1.66	16.01	—	0.005	0.001	-0.029	45
	sexual contacts	undirected	2 810	—	—	—	3.2	—	—	—	265, 266
information	WWW nd.edu	directed	269 504	1 497 135	5.55	11.27	2.1/2.4	0.11	0.29	-0.067	14, 34
	WWW Altavista	directed	203 549 046	2 130 000 000	10.46	16.18	2.1/2.7	—	—	—	74
	citation network	directed	783 339	6 716 198	8.57	—	3.0/-	—	—	—	351
	Roget's Thesaurus	directed	1 022	5 103	4.99	4.87	—	0.13	0.15	0.157	244
	word co-occurrence	undirected	460 902	17 000 000	70.13	—	2.7	—	0.44	—	119, 157
technological	Internet	undirected	10 697	31 992	5.98	3.31	2.5	0.035	0.39	-0.189	86, 148
	power grid	undirected	4 941	6 594	2.67	18.99	—	0.10	0.080	-0.003	416
	train routes	undirected	587	19 603	66.79	2.16	—	—	0.69	-0.033	366
	software packages	directed	1 439	1 723	1.20	2.42	1.6/1.4	0.070	0.082	-0.016	318
	software classes	directed	1 377	2 213	1.61	1.51	—	0.033	0.012	-0.119	395
	electronic circuits	undirected	24 097	53 248	4.34	11.05	3.0	0.010	0.030	-0.154	155
biological	peer-to-peer network	undirected	880	1 296	1.47	4.28	2.1	0.012	0.011	-0.366	6, 354
	metabolic network	undirected	765	3 686	9.64	2.56	2.2	0.090	0.67	-0.240	214
	protein interactions	undirected	2 115	2 240	2.12	6.80	2.4	0.072	0.071	-0.156	212
	marine food web	directed	135	598	4.43	2.05	—	0.16	0.23	-0.263	204
	freshwater food web	directed	92	997	10.84	1.90	—	0.20	0.087	-0.326	272
	neural network	directed	307	2 359	7.68	3.97	—	0.18	0.28	-0.226	416, 421

TABLE II Basic statistics for a number of published networks. The properties measured are: type of graph, directed or undirected; total number of vertices  $n$ ; total number of edges  $m$ ; mean degree  $z$ ; mean vertex–vertex distance  $\ell$ ; exponent  $\alpha$  of degree distribution if the distribution follows a power law (or “—” if not; in/out-degree exponents are given for directed graphs); clustering coefficient  $C^{(1)}$  from Eq. (3); clustering coefficient  $C^{(2)}$  from Eq. (6); and degree correlation coefficient  $r$ , Sec. III E. The last column gives the citation(s) for the network in the bibliography. Blank entries indicate unavailable data.

Le reti sociali e umane, soprattute quelle di una certa dimensione hanno valori positivi e piuttosto significativi, a riprova del fatto che le reti sociali sono assortative.

## Centrality

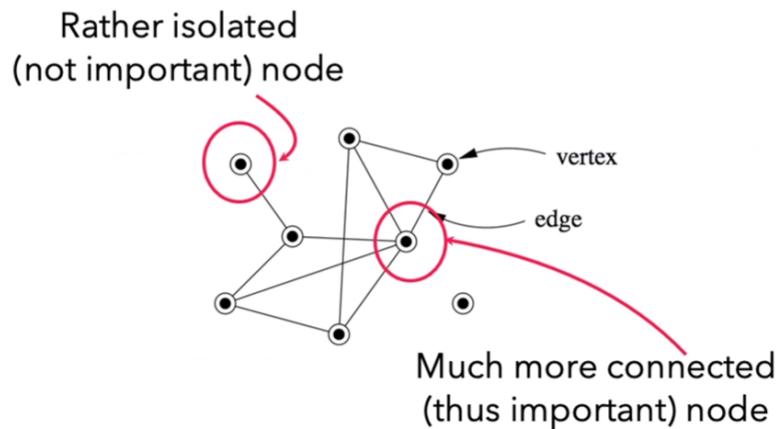
### Roadmap



Gli indici di centralità sono una classe di indici:

## General Concept of Centrality

- Given a node, how "central" (thus, important) is it in the network?



di fatto qualunque numero che mi permetta di fare ranking dei nodi per metterli in classifica.

Lo si fa per permettere di capire l'importanza dei nodi dal meno al più importante. Per reti piccole non serve questo indice in quanto nell'esempio mostrato sarebbe facile produrre la risposta. Servono diverse misure di centralità a seconda del motivo per cui si fa il ranking tra i nodi. A seconda del motivo per cui faccio la classifica tra i nodi devo usare un indice differente.

## Centrality indices

- Degree centrality**
  - A node is central if it is very well connected
- Closeness centrality**
  - A node is central if it is "close" to the other nodes
- Betweenness centrality**
  - A node is central if it is "in between" paths between other nodes

$$c_{CI}(v) = \frac{1}{\sum_{u \in V} \text{dist}(v, u)}$$

sum of path lengths to all other nodes

$$c_B(v) = \sum_{s \neq t \neq v \in V} \frac{\sigma(s, t | v)}{\sigma(s, t)}$$

Number of shortest paths between s and t that go through v

For all possible pairs of nodes (s, t) which are not v

Number of shortest paths between s and t

- Degree centrality:** Il grado dei nodi lo uso per associare un importanza a ciascun nodo. La semantica di questa regola è che un nodo è importante se è molto connesso.
- Closeness centrality:** la vicinanza (closeness) ovvero mi interessa avere una classifica di vicinanza ovvero quanto sono lunghi gli shortest path tra un nodo e tutti gli altri nodi della rete. Matematicamente viene calcolato con la prima formula mostrata sulla destra. Se voglio calcolare la centralità del nodo  $v$  mi prendo tutti gli altri nodi all'interno della rete, in questo caso è chiamato

$v$ . Mi calcolo la distanza tra tutti questi nodi in  $v$  e il nodo  $v$ , faccio la somma e prendo il reciproco.

Non è importante il valore assoluto degli indici ma la classifica, dunque in questo caso la formula prevede di fare il reciproco. L'indice di closeness non si usa molto.

- **Betweenness centrality:** ci dice quanto un nodo è "in mezzo" quando si vuole andare da una parte all'altra della rete. Mi chiedo quanto questo nodo controlla di fatto i percorsi all'interno della rete. Ovvero controllare un gran numero di percorsi per muoversi da una parte all'altra della rete è chiaramente un punto di vantaggio. Ponendoci sul nodo  $v$ , prendiamo un'altra coppia di nodi  $s$  che sta per sorgente e  $t$  che sta per target. Calcolo tutti gli shortest path tra  $s$  e  $t$ . Anche se considero i path più corti, ce ne può essere più di uno con la medesima lunghezza.

$\sigma(s,t)$  è il numero di shortest path che ci sono tra  $s$  e  $t$ . Il numeratore invece è sempre il numero di shortest path tra  $s$  e  $t$  ma solo se attraversano il nodo  $v$ . Questo rapporto è un numero che sta tra 0 e 1.

Se data una particolare coppia  $s$  e  $t$  questa frazione è tanto più uguale a 1, tanto più significa che devo passare per forza per  $v$  per andare da  $s$  a  $t$ . Questo numero mi dà la probabilità che per connettere  $s$  e  $t$  io debba passare attraverso  $v$ .

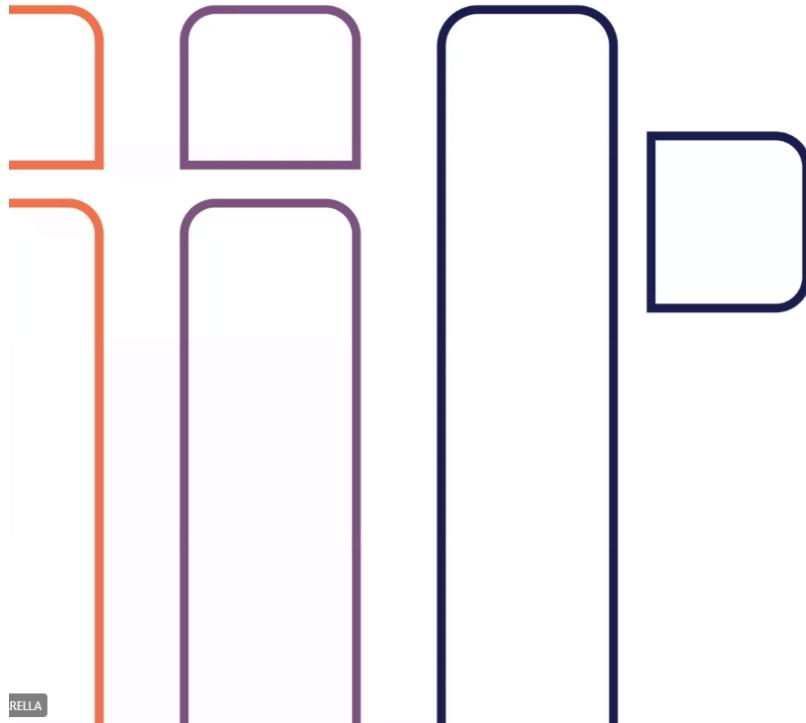
Questa operazione la faccio per tutte le possibili coppie di nodi all'interno della rete e sommo tutti questi indici.

L'indice di betweeness si può calcolare anche sugli edges, ovvero anzichè in riferimento a un nodo, posso calcolarlo in riferimento a un edge.

---

30/06/2023

---



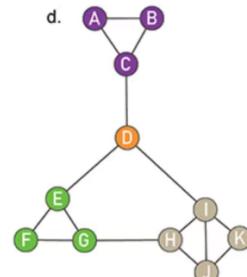
## Complex Network Analysis Community Detection

---

## Communities

---

- ❑ In many cases the structure of network is clustered
  - High clustering coefficient
- ❑ This typically corresponds to the presence of **natural communities** of nodes
  - E.g., social communities in social networks
- ❑ Communities ≠ Connected Components
  - More communities can be present in the same CC
  - Communities do not span across CCs



Le **comunità** sono gruppi di nodi in una rete molto collegati tra di loro ma piuttosto ambiguo da definire formalmente. E' una delle strutture fondamentali quando si analizza una rete non ultimo per motivi legati alla robustezza.

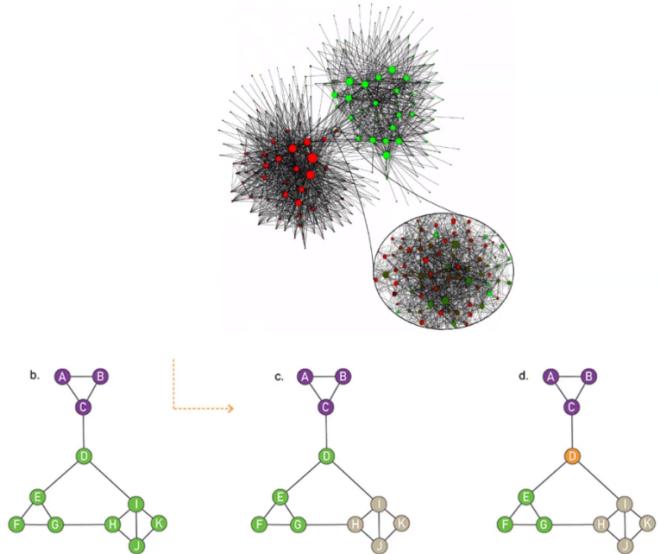
Nella figura in basso a destra abbiamo un unico connected components ma sono presenti 4 comunità differenti.

In generale le comunità si cercano all'interno dello stesso connected components. Non può succedere che una comunità sia a cavallo tra connected components diverse.

## Issues in identifying communities

---

- ❑ Communities can easily be identified in small networks
  - But how can we do it in **large** networks?
- ❑ How to identify the "**best**" split of nodes in communities?



le comunità sono gruppi di nodi che sono connessi tra di loro **molto di più** rispetto a quante sono le connessioni di ciascuno dei nodi della comunità con il mondo esterno. Basti pensare alla comunità di amici stretti che ognuno ha, che è un insieme di connessioni molto più forti rispetto alle conoscenze di altre persone (colleghi, amici dello sport per esempio).

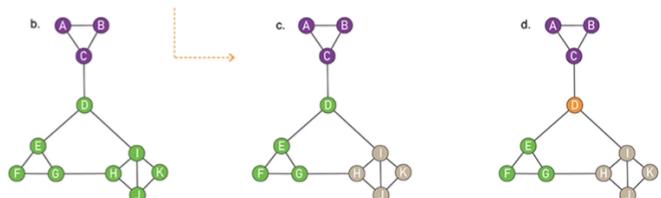
A seguito di questa ambiguità nella definizione, non sempre le comunità vengono identificate in modo univoco, si veda l'esempio in cui abbiamo 3 diversi insiemi di comunità identificate (b,c,d).

L'area della community detection è un area di ricerca attiva dove ci si occupa di trovare algoritmi

efficienti per trovare le comunità su larga scala.

## Partitions and communities

- ❑ In each of these case we have a **partition** of the network in groups
  - ❑ Partition = way of grouping nodes
  - ❑ 2, 3, 4 groups respectively
- ❑ What **partition** corresponds to the **best community structure** of the network?

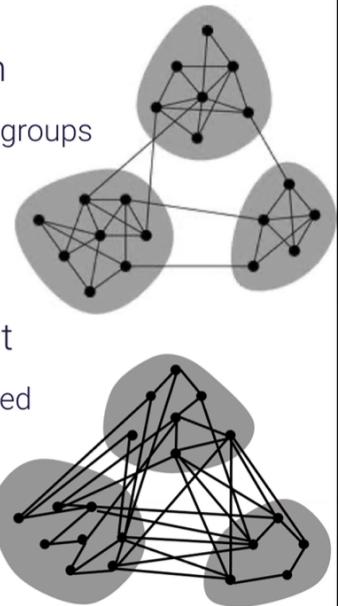


Trovare le comunità significa **partizionare** la rete ovvero identificare una partizione di nodi ovvero un modo di raggruppare i nodi che corrisponda il più possibile con la definizione data prima.

## Modularity

- ❑ **Index** defined on a specific partition
- ❑ measures how “**far**” that partition is from a **random** assignment of the same links over the same nodes
- ❑ **High modularity** ⇒ far from random assignment ⇒ indication of a **good split** in “natural” communities

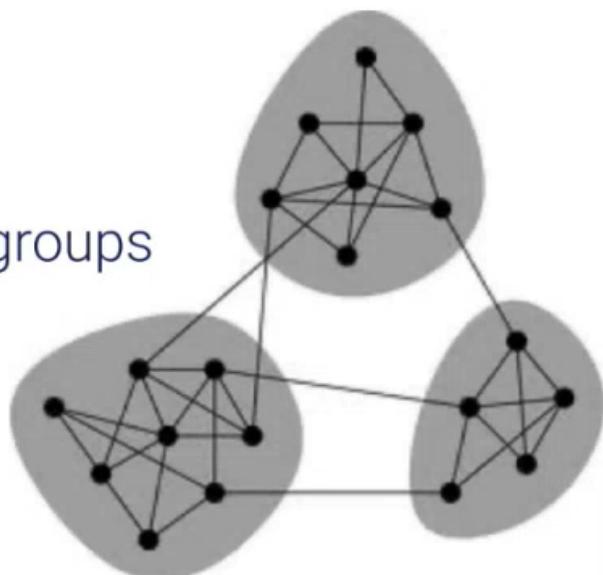
- ❑ “Candidate” partition
  - ❑ nodes divided in 3 groups
- ❑ Random assignment
  - ❑ Same links, assigned randomly



**L'indice di modularità** è un numero che mi dice che una volta che ho assegnato i nodi a una certa partizione candidata, prendo un'altra partizione candidata e ne confronto gli indici. L'indice di modularità per come è definito tanto è più alto quanto più il partizionamento che sto considerando si avvicina a quell'idea di comunità definita prima. Ovvero se definisco due partizionamenti candidati all'interno della mia rete e calcolo le modularità della rete rispetto a questi partizionamenti, il partizionamento per cui ottengo l'indice di modularità più alta vuol dire che rispecchia meglio la proprietà vista prima ovvero nella rete esistono molti più link all'interno dei nodi dei gruppi rispetto a quanto ne esistano all'esterno. L'idea dell'indice di modularità la diamo analizzando le figure in alto: il partizionamento candidato viene rappresentato usando le 3 bolle grigie in alto:

## ❑ “Candidate” partition

- ❑ nodes divided in 3 groups



Nella rete in alto sto considerando il partizionamento candidato con le 3 bolle grigie ovvero proviamo a supporre che le comunità siano formate sulla base di queste 3 bolle grigie. Dato questo partizionamento possiamo definire i link in 2 categorie:

- **intracommunity**: collegano nodi nello stesso gruppo nel partizionamento considerato
- **intercommunity**: collegano nodi di gruppi diversi nel partizionamento che stiamo considerando

Qui abbiamo 5 link intercommunity e tutti gli altri sono intracommunity. L'indice di modularità ci dice quanto è distante la distribuzione dei link rispetto alla figura in basso, generata levando tutti i link della rete, prendendo un link alla volta e usandolo per collegare 2 nodi a caso nella mia rete. Ottengo la rete in basso, ovvero una rete equivalente (stessi nodi e numero di link) ma la distribuzione dei link sui nodi è randomica. L'indice di modularità mi dice quanto relativamente alla proporzione intra/inter link le 2 reti sono distanti.

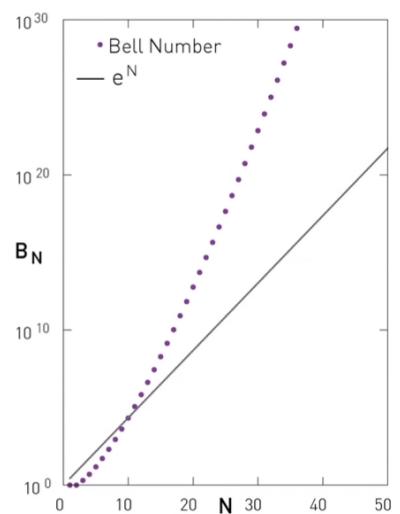
! L'intuizione è che più questo indice è alto più le 2 reti sono distanti ovvero se prendo come buona l'ipotesi che quelle cerchiate in grigio siano le mie comunità, se **l'indice di modularità è alto** significa che la rete in alto è molto diversa dalla rete in basso ovvero la distribuzione dei link inter/intra community è molto distante da un assegnamento random, ovvero ho trovato un partizionamento dei nodi per cui è difficile riottenere quella distribuzione di link assegnando i nodi in modo puramente casuale.

La differenza massima l'avremmo se nella rete originale avessi solo 2 link che mi collegano le comunità tra di loro (nel nostro esempio SOLAMENTE 2 link intercommunity e tutti gli altri link intracommunity così da avere un componente连通的) in questo caso avremmo la massima modularità dunque la massima distanza rispetto alla configurazione (randomica) che otteniamo con il grafo in basso.

? Come si usa questo concetto di modularità?

## Complexity

- ❑ Possible algorithm
  - ❑ Identify all possible partitions
  - ❑ Compute modularity
  - ❑ Pick the partition with highest modularity
- ❑ Feasible?
  - ❑ NO
  - ❑ Number of partitions increases **exponentially** (or more) with the size of the network
  - ❑ We need a way to quickly identify "good candidate partitions"



Posso calcolarmi tutti i possibili partizionamenti, peccato però che computazionalmente non è trattabile in quanto nel grafico i pallini sono il numero di comunità che si generano in una rete di dimensione  $N$ , dunque il numero di partizionamenti da controllare cresce esponenzialmente.

Sono necessarie delle tecniche euristiche, ovvero con un problema matematicamente non trattabile devo trovare una funzione che mi ottimizza la soluzione (in questo caso la modularità).

? Come faccio a trovare il sottoinsieme di partizionamenti su cui lavorare per evitare di farlo su tutti? Posso inventarmi un algoritmo che mi garantisce con BUONA APPROSSIMAZIONE che in questo sottoinsieme di partizionamenti che considero ci sia anche un sottoinsieme con modularità massima che è quello che mi interessa davvero.

Gli algoritmi di community detection differiscono tra di loro nel modo in cui selezionano le partizioni candidate su cui andare a calcolare la modularità, dimostrando che una certa euristica è migliore o peggiore rispetto a quelle che già si conoscono in letteratura.

## Intuitive definition of communities and community detection

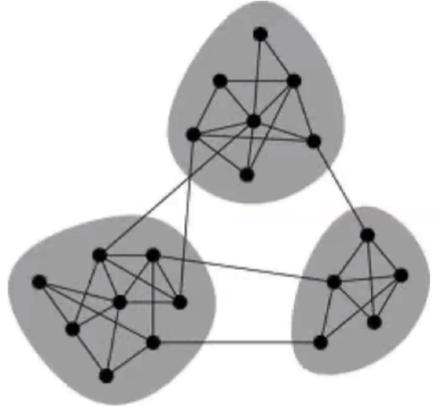
### □ “Density”-based definition

- Nodes have more links towards other nodes of their community than towards nodes of other communities

### □ ⇒ there are relatively “few” links connecting different communities

### □ Community detection

- If one identifies those “few” links and removes them, then one should be able to highlight the “natural” community structure of the network
- We need a corresponding index to rank links



Uno dei primi e più famosi è basato sul concetto di **densità** ovvero una misura di quanto, dato un certo gruppo di nodi, sono collegati al loro interno rispetto ai collegamenti esterni.

💡 L'idea è quello di cercare i link intercommunity. I link intercommunity sono pochi e soprattutto ipotizzando che questi 3 gruppi di nodi siano quelli reali, se elimino i link intercommunity, ottengo le 3 comunità come sottogruppi di nodi isolati. Se riuscissi dunque a identificare i link intercommunity e li eliminassi e mi calcolassi i componenti connessi del grafo che mi rimane che è diverso da quello originale in quanto sto rimuovendo dei link, a un certo punto dovrei riuscire ad avere isolato quelle che sono le comunità all'interno della mia rete, perchè quelli sono i componenti connessi.

Supponiamo infatti di nominare i link con indici da 1 a 5. Elimino il primo link e ho ancora il componente连通的, elimino il link 2 e ho ancora un componente连通的, continuo fin quando non elimino tutti e 5 i link intercommunity e ottengo 3 componenti连通的 che guarda caso sono proprio le mie 3 comunità.

Il problema è come capire quali sono i link intercommunity.

Rispetto agli indici già visti, se volessimo calcolare un indice sugli edge che aiuta a capire quali sono i link intercommunity, quale dovremmo usare? Il betweeness.

Se consideriamo la betweeness calcolata sugli edge, e lo calcoliamo per tutti i link all'interno di questa rete, questi 5 link intercommunity hanno un indice di betweeness più alto di tutti gli altri link perchè quando mi voglio spostare da un nodo all'altro all'interno della mia comunità, ogni volta che la coppia di nodi sta su 2 comunità diverse devo per forza passare da uno di quei link intercommunity che sono pochi quindi ci saranno tanti path che passano su quei link intercommunity. Se prendo un link intracomunity saranno di meno i path che insistono su quel link. Quindi l'indice di betweeness calcolato su gli edge sembra essere un ottimo candidato per trovare i link intercommunity. Posso infatti:

- calcolarmi tutti i link di betweeness centrality dei link
- ordino i link per betweeness centrality decrescente
- prendo il link con betweeness centrality massima e lo elimino dalla rete

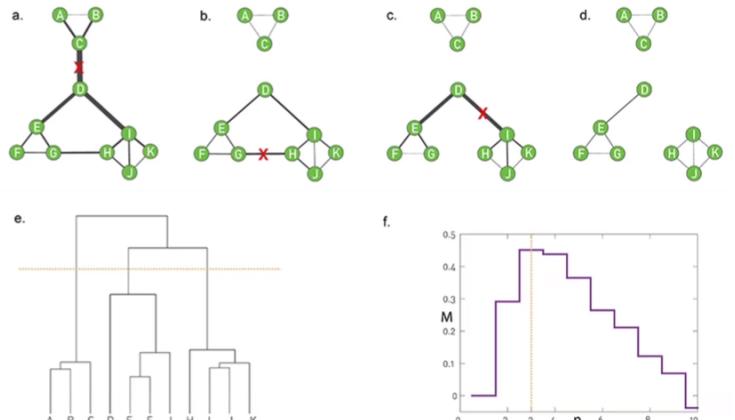
- dal nuovo grafo calcolo i componenti connessi e se non ne ho più uno unico, ho un partizionamento candidato ovvero ho identificato un certo gruppo di nodi
- assumo che questa possa essere la struttura comunitaria della mia rete e sulla rete originale ne calcolo l'indice di modularità
- ritorno poi nel grafo in cui stavo rimuovendo i link e continuo a rimuoverli fin tanto che non ottengo un diverso insieme di componenti connessi
- ricalcolo la modularità secondo il nuovo partizionamento e ottengo un ulteriore indice di modularità

Questo algoritmo non esplora tutti i partizionamenti candidati ma identifica dei partizionamenti candidati rimuovendo gli edge a betwenees massima fin tanto che la struttura di questi candidati non si modifica e a quel punto calcola un indice di modularità:

## Girvan-Newman Community Detection

---

- Compute the **betweenness**  $g(e)$  of each link
- Remove the link with the **largest** betweenness
  - In case of a tie, choose one link randomly
- Compute the **modularity** of the obtained partition
- Recalculate the betweenness of each link for the altered network
- Repeat until all links are removed
- Partition with the **highest modularity** is the community structure



❓ Quando ci fermiamo nel calcolare le partizioni candidate?

Quando la modularità non cresce più. E' stato dimostrato che questo è un buon criterio di stop: via via che rimuovo i link a betweenness alta separo i componenti che corrispondono alle mie comunità dunque sto andando nella direzione di trovare dei partizionamenti a modularità alta perché via via che rimuovo i link a betweenness alta ovvero quelli intercommunity sto separando i componenti che corrispondono alle mie comunità. Si generano dunque partizioni che sono artificiali, l'indice di modularità di queste sottopartizioni delle comunità reali tenderà a diminuire man mano che continuo a partizionare le comunità generandone altre artificiali. Partiamo con la rete nel caso a. Il link con la betwenees più alta di tutti è quello in rosso. Eliminandolo ottengo 2 componenti connessi. Calcolo la modularità del grafo originale che registro nel grafico in basso a destra. Ora il nuovo link a betweenness massima è tra G e H, eliminandolo e calcolandolo la modularità non cambia nulla. E così via, questo **algoritmo di Girvan-Newman per l'anomaly detection** si ferma quando la curva inizia a scendere. Si tratta di un'euristica in quanto tolgo via via e mi fermo quando l'indice di modularità inizia a scendere.

## Observations

---

- ❑ Complexity of GN is  $O(N^3)$ 
  - ❑ Optimised algorithms exist to reduce complexity
- ❑ GN is a heuristic
  - ❑ Does not guarantee that all partitions are tested, so in principle there could be partitions with highest modularity
    - ❑ But this happens very rarely
- ❑ Many different definitions of communities exist
  - ❑ And many different community detection algorithms too

Con la community detection abbiamo finito gli indici necessari per analizzare la rete. Tutti gli indici presentati permettono di avere una fotografia della rete da punti di vista diversi.

## Robustness



# Complex Network Analysis

## Robustness

Andrea Passarella

a.passarella@iit.cnr.it



Istituto  
di INFORMATICA  
e TELEMATICA



Consiglio Nazionale  
delle Ricerche

ANDREA PASSARELLA

### How to measure robustness

- Robustness = persistence of the Giant Component
  - "up to which point removing nodes/edges the Giant Component still exists?"
- Alternative measure: increase of the average path length
- Types of attacks
  - Random attacks (failures)
    - nodes/edges are selected with a uniform distribution and removed
  - Targeted attacks (attacks)
    - nodes/edges are ranked according to some index, and removed starting from the "most important"
- Attacks over time
  - Simultaneous attacks
    - Rank nodes/edges in the original network, remove a fraction  $\rho$  simultaneously
  - Sequential attacks
    - while (removed <  $\rho$ ): rank\_nodes(); remove\_first(); removed++;

**Si tratta della persistenza del Giant Component al variare del numero di nodi o edge che eliminano dalla rete.**

Se rimuovo ad esempio un 20% dei nodi della rete, di quanto si è ridotto il giant component della rete?

Se prendiamo il giant component di una rete e iniziamo a eliminare dei nodi o link, questo potrà rimanere uguale o diminuire.

L'indice di robustezza che consideriamo è, dopo aver distrutto una certa percentuale di nodi o link, di quanto si è ridotto il giant component in percentuale.

❓ Come possiamo identificare i nodi o link da distruggere?

Si categorizzano gli attacchi secondo 2 dimensioni:

- per **tipologia**:
  - negli **attacchi random** non si usa nessun criterio, si attacca un nodo o un edge a caso

- negli **attacchi targeted** ho una strategia in base a un indice di importanza dei nodi o edge da attaccare così da definire la stima del danno che riesco ad arrecare
- per **sviluppo nel tempo**:
  - simultanei**: l'indice di rank è calcolato una volta sola nella rete originale e viene rimossa solo una frazione  $p$  della rete
  - sequenziali**: viene ricalcolato l'indice ogni volta che un nodo/link viene rimosso

Data una strategia di attacco, dunque assodato che si usa un attacco di tipo targeted, in generale sono più efficaci gli attacchi sequenziali.

Il **critical threshold** è la percentuale di nodi rispetto a una particolare strategia di attacco che devo eliminare tale per cui la frazione del giant component che sopravvive è trascurabile ovvero prossima a zero.

## Parameters for robustness

---

### Critical threshold (%): $f_c$

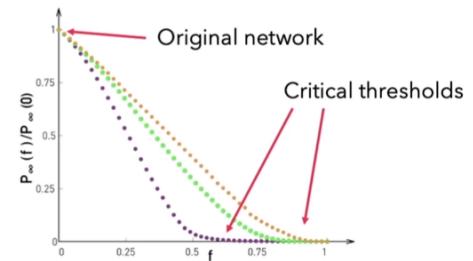
- Removing more than  $f_c$  nodes destroys the Giant Component

### Size of the GC

- relative to the size of the GC in the original network

### Probability that a random node is part of the GC

- with respect to the probability in the original network



Ogni curva corrisponde a un particolare attacco, sulle x ho una percentuale di nodi che ho distrutto mentre sulle y ho una frazione di giant component che dopo aver distrutto quella percentuale di nodi mi sopravvive rispetto alla dimensione del giant component originale.

Sulle x ho la percentuale di nodi distrutti mentre sulle y la percentuale di nodi del giant component che sopravvive.

L'attacco è tanto più efficace quanto più questa curva scende rapidamente.

Qual'è la percentuale di nodi tale per cui il componente连通的 che mi rimane è prossimo a 0?

Il viola è quello più efficace perchè il suo critical threshold è prossimo al 50% in quanto gli basta distruggere circa metà nodi per essere efficace.

Questi critical threshold si possono calcolare secondo alcuni modelli visti durante il corso e ci sono dei valori ben precisi:

## Critical Thresholds - Random Failures

- For random graphs (Erdos-Renyi)  $f_c^{ER} = 1 - \frac{1}{\langle k \rangle}$
- For scale-free networks ( $2 < \gamma < 3$ )  $f_c^{SF} = 1$
- For scale-free networks ( $3 < \gamma < 4$ )  $f_c^{SF} = 1 - \frac{1}{\frac{\gamma-2}{\gamma-3} k_{min} - 1}$

per i random graph quando considero attacchi di tipo random il critical threshold è  $1 - 1/(\text{grado medio})$ . Questo vuol dire che i grafi ER anche se li attacco in modo random esiste una percentuale di nodi oltre la quale il giant component diventa trascurabile ovvero ho distrutto la rete.

Per reti scale free vere il critical threshold è uguale a 1, dunque se ho 2 grafi con lo stesso numero di nodi e di link però uno ha una distribuzione ER e uno scale-free, la seconda è più robusta agli attacchi random. Via via che l'atteggiamento power law diventa sempre meno evidente, ovvero il gamma cresce, il critical threshold diminuisce e asintoticamente diventa uguale ai grafi ER.

❓ Come mai le reti scale-free con  $\gamma$  basso ovvero reti con una distribuzione di tipo power law sono molto robuste ad attacchi di tipo random? Per la presenza degli HUB! In quanto gli hub sono pochi e sono gli unici a mantenere la rete, se però non ho un criterio, è poco probabile che randomicamente colpirò un hub. Nei random graph non ho hub dunque tutti i nodi sono più o meno equivalenti tra di loro e non hanno dunque gli hub a proteggerli.

❓ Perchè le reti power law con un  $\gamma$  tra 2 e 3 sono robuste contro attacchi random?

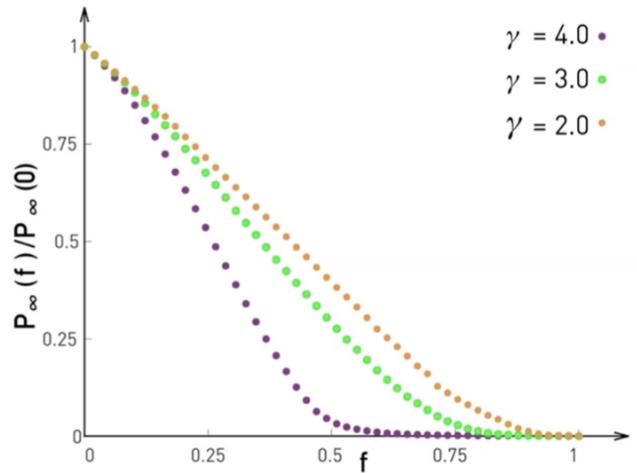
La presenza di hub che sono pochi che tengono insieme la rete, per farla male devo sceglierli, ma se non ho un criterio e colpisco randomicamente è molto improbabile che distruggerò l'hub.

Invece nei random graph non ho hub dunque tutti i nodi sono equivalenti tra di loro.

Queste 3 curve si riferiscono al medesimo attacco di tipo random fatto però su 3 reti power law con una  $\gamma$  diverso e quella più robusta di tutte è quella gialla con  $\gamma = 2$ .

## Random Failures - Scale Free Networks

- SF networks are
  - extremely robust to random failures
    - As  $f_c \sim 1$
- They become less and less robust as the exponent increases
- Why?
  - robustness is because the GC is "kept together" by the few hubs with very high degree
  - these are very rare, and therefore difficult to "destroy" via random choice with uniform distribution (all nodes are destroyed with the same probability)
  - as the exponent increases, hubs are smaller, and therefore they "increasingly need also additional nodes" to keep the GC together



In caso di reti reali come le power grid, avendo una distribuzione esponenziale e non power law, se attaccata in maniera random è meno robusta di tutte le altre reti di tipo power law.

## Random Failures - real networks

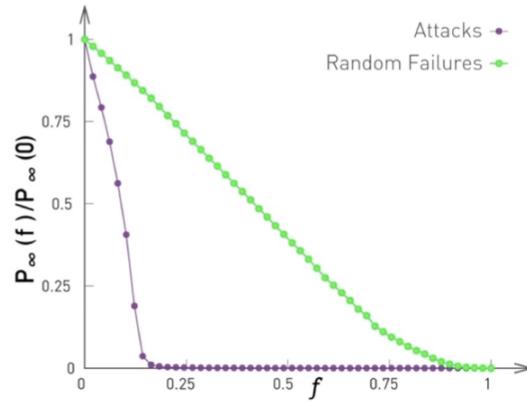
Network	Random Failures (Real Network)
Internet	0.92
WWW	0.88
Power Grid	0.61
Mobile Phone Calls	0.78
Email	0.92
Science Collaboration	0.92
Actor Network	0.98
Citation Network	0.96
E. Coli Metabolism	0.96
Protein Interactions	0.88

Se invece effettuo degli attacchi targeted e come criterio di attacco prendo il grado dei nodi, una rete scale-free diventa molto fragile.

Abbiamo la stessa rete attaccata secondo due modi diversi. Le reti scale-free sono dunque molto robuste rispetto ai fallimenti ma estremamente vulnerabili rispetto agli attacchi targeted sul grado. Se vogliamo proteggere gli hub con un investimento economico sostanzioso c'è da dire che essendo pochi conviene farlo, in quanto in tal caso diventa difficile andare a distruggerli.

## Critical Threshold - Targeted Attacks

- ❑ Attack the **highest-degree** nodes
  - ❑ while True: `n_h = highest_degree_node(); kill(n_h);`
- ❑ Difference between random and targeted attacks for **SF** networks
- ❑ Why?
  - ❑ attacks targeted to the high-degree nodes kill those few nodes that "keep the GC together"
  - ❑ it is sufficient to kill a few to make the GC break apart



Se uno guarda i critical threshold non relativamente agli attacchi random ma relativamente agli attacchi target al grado, le reti con estrema robustezza ai random failure diventano molto vulnerabili agli attacchi targeted.

## Critical Thresholds - Real Networks

Network	Random Failures (Real Network)	Attack (Real Network)
Internet	0.92	0.16
WWW	0.88	0.12
Power Grid	0.61	0.20
Mobile Phone Calls	0.78	0.20
Email	0.92	0.04
Science Collaboration	0.92	0.27
Actor Network	0.98	0.55
Citation Network	0.96	0.76
E. Coli Metabolism	0.96	0.49
Protein Interactions	0.88	0.06

💡 In qualche maniera abbiamo due comportamenti che sembrano mutuamente esclusivi, non possiamo fare in modo che se ho una rete scale free ho una rete più robusta rispetto agli attacchi target di come sarebbe naturalmente? Ovvero è possibile cambiare la struttura della rete in modo da renderla più robusta di come sarebbe naturalmente?

L'indice che si vuole massimizzare è quello mostrato in fondo alla slide.

## Increasing Robustness

---

- Intuition: increasing robustness of SF graphs to attacks
  - make sure that not only the "big hubs" keep the GC together
- Intuition: increasing robustness of RG graphs to failures
  - Increase degree "diversity" (make sure there are larger hubs)
- Is it possible to make a network simultaneously robust to both attack and failures?
- Typical performance target
  - maximise the sum of the critical thresholds in the two cases  $\max \{ f_c^{rand} + f_c^{targ} \}$

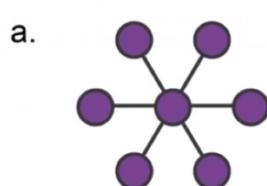
Un modo banale di fare questa operazione potrebbe essere quello di aggiungere più link. Pensiamo a cosa succede tra la rete **a** e la rete **b**. La rete **a** è una rete con un hub e tanti link a grado molto basso, io potrei trasformarla nella rete **b** in cui aumento il grado dei nodi che non sono hub e in questo modo aumento la robustezza. Avendo aumentato il numero di link significa aver aumentato il grado medio ma anche aver speso magari un ingente somma di denaro

Dato un numero di link che posso inserire in una rete come possiamo renderla il più possibile robusta?

## Increasing Robustness

---

- Easy way: increasing the number of links
  - to avoid that only the hubs keep the GC together
  - drawback: **cost** (building links may be very expensive)
- How to increase robustness **without** increasing the average degree?



$$\langle k \rangle = 12 / 7$$



$$\langle k \rangle = 24 / 7$$

Esiste un altro risultato che dice che per reti generiche ovvero che non hanno una particolare distribuzione del grado, il critical threshold per le failure è uguale a questa formula:

## Maximise Variance

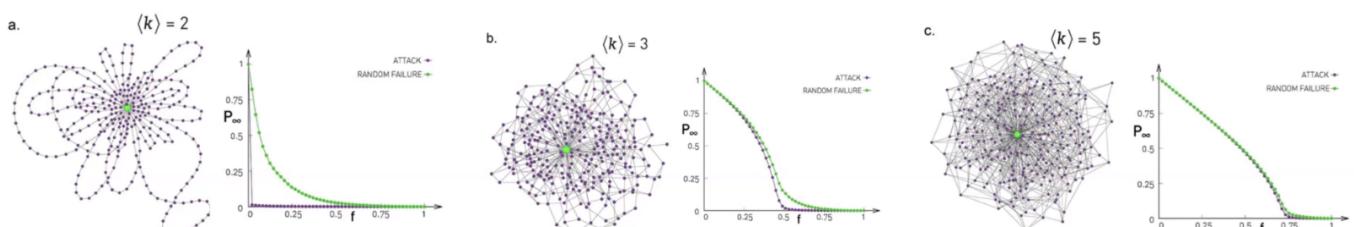
- ❑ For generic networks, the critical threshold to failures is  $f_c = 1 - \frac{1}{\frac{\langle k^2 \rangle}{\langle k \rangle} - 1}$
- ❑ As  $\langle k \rangle$  needs to remain constant, the way to maximise  $f_c$  is to maximise  $\langle k^2 \rangle$ 
  - ❑ i.e., to maximise the variance of degrees in the GC
- ❑ Bimodal distribution
  - ❑ a fraction  $r$  of nodes has maximum degree, a fraction  $(1-r)$  has minimum degree

il  $k^2$  sarebbe il **momento secondo** ovvero la varianza dei gradi, quello che ci dice è che per incrementare la robustezza alle random failure la rete dovrebbe avere più varianza possibile nel grado dei nodi. Questo è un trucco infatti che usano le reti scale free in quanto hanno una grande varianza nella distribuzione dei nodi. Massimizzare la varianza non basta però perché ottengo una rete robusta rispetto agli attacchi random ma non rispetto a quelli target.

Dopo un po di matematica la rete che è massimamente robusta rispetto agli attacchi ha una certa frazione di nodi che ha un degree massimo che si può calcolare (dipende dal budget a disposizione) e tutti gli altri nodi hanno un grado minimo uguale che si può calcolare. Dunque si trova matematicamente che la rete massimamente robusta non ha una distribuzione esponenziale ne power law ma una distribuzione con soli 2 valori possibili: uno massimo e uno minimo e questi dipendono da quanto è grande il grado medio che possiamo permetterci.

## Networks with Maximal Robustness

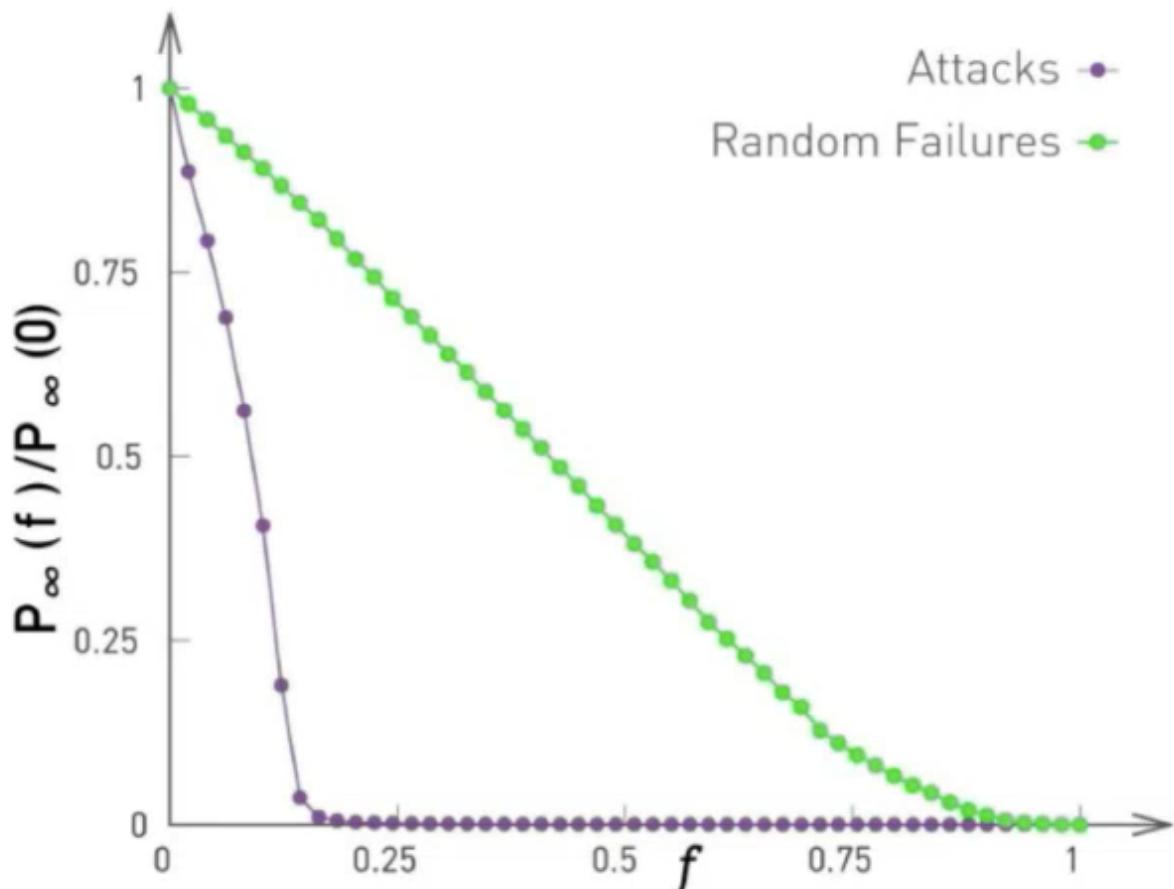
- ❑ How far we can go depends on  $\langle k \rangle$ 
  - ❑ very small  $\langle k \rangle$  lead to vulnerable networks
  - ❑ even moderate  $\langle k \rangle$  can result in fairly robust networks



I nodi a **grado massimo** deve essere **UNO SOLTANTO** mentre quelli a grado minimo tutti gli altri. Nel caso a sinistra ho solo 2 link da budget in media da aggiungere. La rete massimamente robusta è

quella con un hub soltanto e tutto il resto con un grado minimo. Questa rete non è molto robusta in quanto abbiamo poco budget, ovvero pochi link a disposizione. Il caso  è quello migliore.

In questo caso non abbiamo trovato una rete con critical threshold uguale a 1 come una rete power law, quindi la stiamo indebolendo contro gli attacchi random ma la stiamo rafforzando contro gli attacchi targeted, perchè rispetto a questi è vero che il critical threshold è sempre uguale a 0.5 in tutti i casi ma una rete power law equivalente ha un critical threshold contro gli attacchi targeted pari al 10%:



dunque un ottimo risultato.

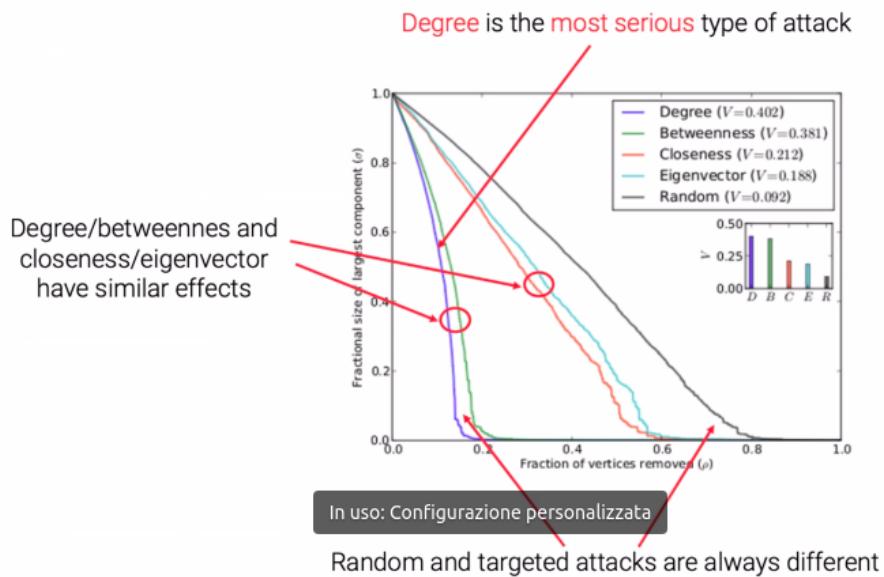
? Qual'è il trucco per cui questa distribuzione dei gradi funziona? Perchè avere un solo hub funziona? Perchè così si prende il meglio dei due mondi in quanto io ho un solo hub, ovvero massimizzo la varianza e mi tiene insieme la rete e mi da la robustezza rispetto ai random attacks in quanto è solo uno e trovarlo è molto improbabile. Ma gli altri link che posso spendere non li investo per costruire degli altri hub ma per dare maggiore robustezza agli altri nodi in modo distribuito, è come se stessi costruendo una rete in cui c'è un superman ovvero l'hub però poi tutti gli altri sono più o meno uguali in quanto avere una rete con tanti nodi tutti uguali tra loro è il trucco per avere una rete robusta rispetto agli attacchi targeted perchè se levato l'hub tutti gli altri sono equivalenti, se ne distruggo un nodo questo può essere rimpiazzato facilmente.

## Robustness and Complex Network Indices

- ❑ Types of attacks
  - ❑ Random attacks (failures)
    - ❑ nodes/edges are selected with a uniform distribution and removed
  - ❑ Targeted attacks (attacks)
    - ❑ nodes/edges are ranked according to some index, and removed starting from the "most important"
- ❑ Targeted attacks can be directed towards many ranking indices
  - ❑ Degree
  - ❑ Betweenness
  - ❑ Closeness
  - ❑ Eigenvector centrality
  - ❑ ...

I targeted attacks si possono fare rispetto a tanti indici di centralità ovvero quando definisco un indice per fare un attacco definisco una classifica dei nodi sulla base della quale l'attaccante riesce a massimizzare il male prodotto alla rete. In linea di principio qualunque indice numerico che mi permetta di creare un ranking dei nodi può essere usato per un targeted attack.

## Typical Behaviour - SF networks



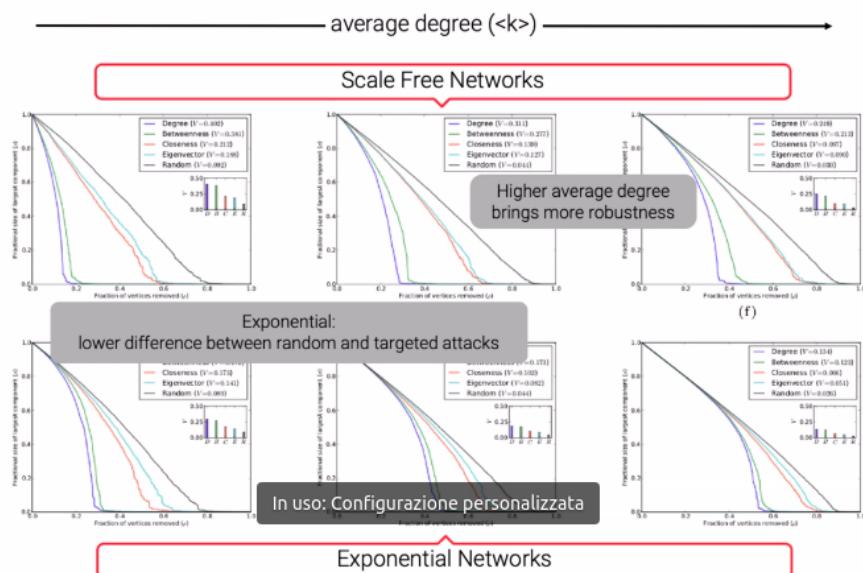
Se prendiamo questa rete che è una power law, se lanciamo attacchi random e targeted su indici mostrati prima, otteniamo queste curve.

1. Al di là del fatto che si tratta di una rete particolare, sulle reti power law, gli attacchi random e qualunque attacco targeted sono diversi. Esiste una differenza nella critical threshold fra le curve targeted e random.
2. Gli attacchi targeted inoltre non sono tutti uguali. Quelli basati sulla degree e betweenness sono simili. Quelli basati su closeness e eigenvector sono simili. **Gli attacchi al grado sono tra i più**

**distruttivi che si possono fare.** Questo è intuitivo in quanto una delle politiche di difesa può essere quella di non far sapere all'esterno le informazioni necessarie per creare questi indici. In una rete però non far sapere quale è il grado dei nodi è molto difficile proprio perché molto spesso l'informazione su qual'è il grado dei nodi è il motivo per cui si costruisce una rete. Pensando a Facebook o Twitter, il grado corrisponde alla importanza e al valore economico degli account che è un'informazione di dominio pubblico.

In generale tra reti a coda leggera e coda pesante come abbiamo visto più volte, su reti scale free le differenze tra attacchi random e targeted sono più marcati che nei corrispondenti casi di reti esponenziali.

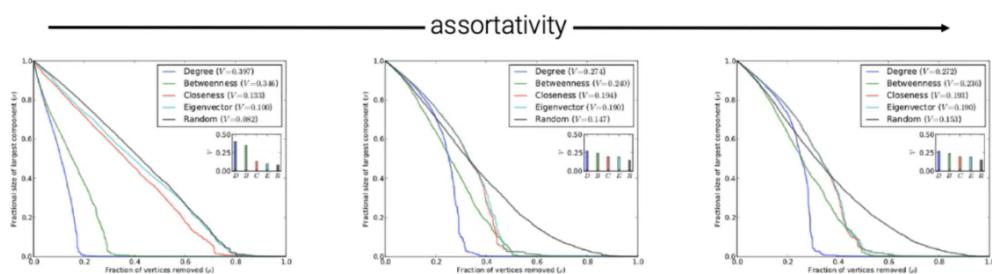
## SF vs Exponential Networks



Su reti scale free esponenziali, benché ci sia una differenza tra attacchi random e targeted, la differenza tra le curve tende ad essere minore in reti esponenziali rispetto a reti scalefree. Inoltre aumentando il grado aumenta la robustezza in quanto aumentano tutti i valori della critical threshold e questo non stupisce in quanto se si vuole aumentare la robustezza di una rete abbiamo detto che conviene aumentare il numero di link.

## Attacks and Assortativity

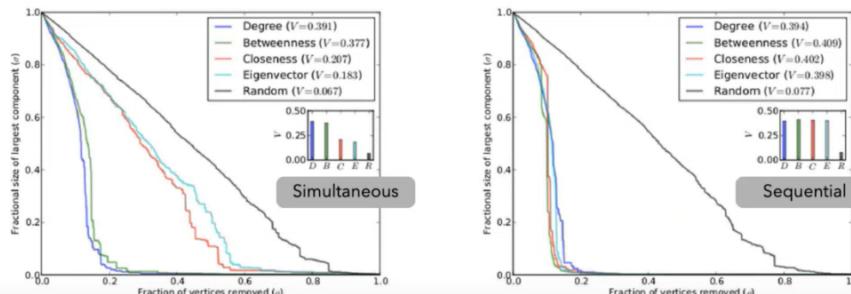
- Assortative networks are more robust to targeted attacks
  - hubs tend to form a cohesive group of nodes
  - the GC is more robust to removal of single hubs



Le reti assortative (quando i nodi a grado alto sono collegati tra di loro) avendo il core in nodi importanti sono più robuste.

## Simultaneous Vs Sequential Attacks

- All types of attack become equivalent to degree-based attacks
- The network is much more vulnerable
  - because the best node to attack is recomputed after each step
- But this is a much more costly attack
  - need to update the state of the network and re-compute the ranking at each step



Gli attacchi sequenziali sono più distruttivi in quanto ogni volta ricalcolo la visione dei nodi critici da attaccare. Le due curve di attacco alla degree e betweenness nel caso di attacco sequenziale o simultaneo sono simili, mentre per altri indici le curve possono essere molto diverse vuol dire che se io mi calcolo l'indice del grado o della betweenness una volta sola sul grafo originale, non sarà il ranking esatto che posso avere con un approccio sequenziale però è un ottima approssimazione.

## Module-based attack

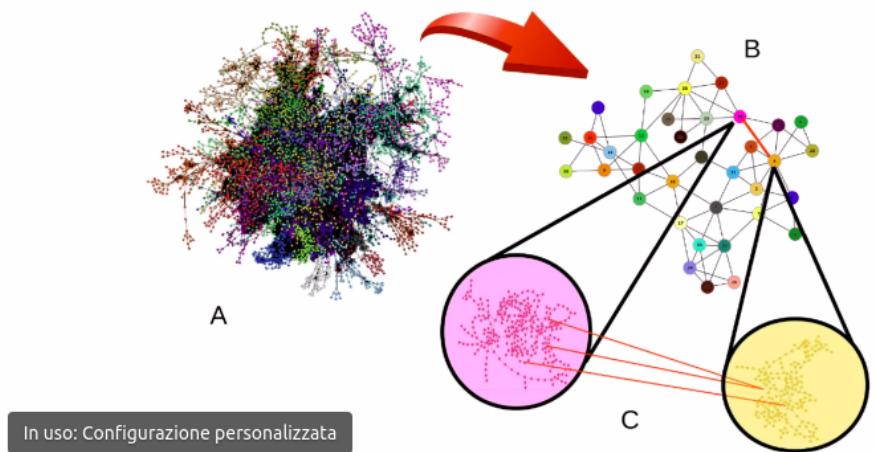
Gli attacchi visti fin ora usano tutti gli indici di centralità dei singoli nodi come indici di ranking.

**Il module based attack punta a reti fortemente strutturate come comunità**, dove isolando le comunità non ci interesserà più distruggere i nodi al suo interno.

## Module-based Attacks

### ❑ Recipe

- ❑ Look for community structures
- ❑ kill nodes (or edges) connecting different communities



Il valore delle reti infatti è quello di connettere il maggior numero possibile di nodi distanti tra di loro ovvero di avere un giant component il più esteso possibile.

💡 Se isoliamo le comunità non ci interessa tanto di andare a distruggere la connettività all'interno della comunità perché mi basta isolare una grande comunità rispetto al resto della popolazione della rete.  
Sulla base di questa idea si possono costruire degli algoritmi di attacco di questo tipo:

## Module-based Attacks: Algorithm

1. Extract communities using a heuristic detection algorithm (see [S1 Text](#) for details).
2. Choose either to attack nodes or edges.
3. Make a list with the nodes (or edges) that participate in intercommunity connections.
4. Sort the list according to (node or edge) betweenness centrality in descending order.
5. Delete nodes (or edges) one by one, starting from the first in the list.
6. While focusing on node removal, once a node from a link between two communities is deleted, its counterpart is skipped from the list (there is no need to remove it), unless it also participates in other intercommunity connections.
7. The attack is always restricted to the largest connected component of the network. In other words, if at some point the In uso: Configurazione personalizzata does not belong to the remaining largest connected component that node (edge) is skipped.

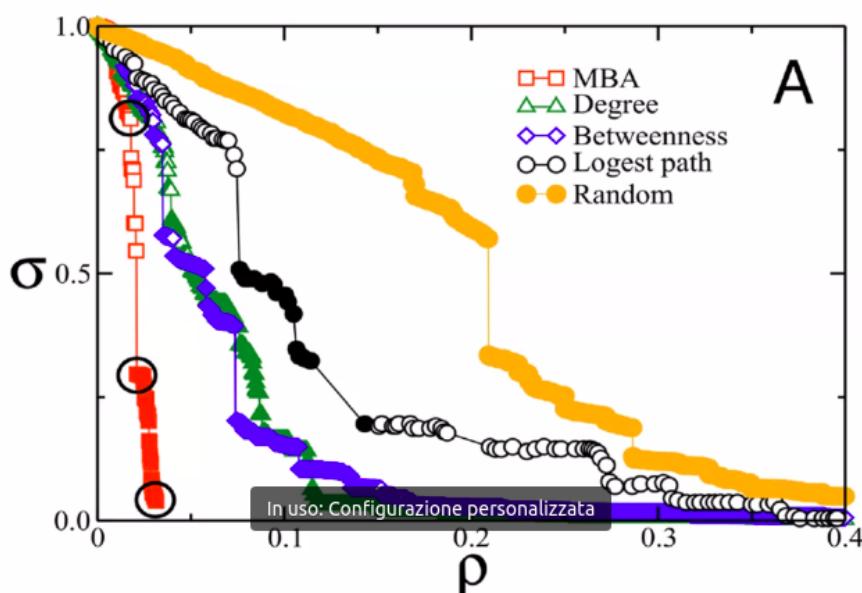
Questo algoritmo permette di estrarre le comunità con un qualunque algoritmo di community detection, si prendono gli edge (si possono considerare anche i nodi volendo) che sono intercommunity, **soltanto** per questi edge calcolo la betweenness centrality, comincio ad eliminare gli edge a partire da quello con la betweenness centrality più alta, (il punto 6 è un ottimizzazione nel caso di attacco ai nodi), l'attacco si fa solamente al largest connected component in quel momento.

Dunque se eliminando l'edge ho disconnesso un pezzo di rete, quel pezzo di rete non mi interessa più, passo a guardare solamente il giant component rimanente.

Dunque si tratta si di un attacco molto costoso in quanto prevede di calcolare la betweenness centrality,

ma va calcolato questo indice solo su una porzione degli edge.

## Effect

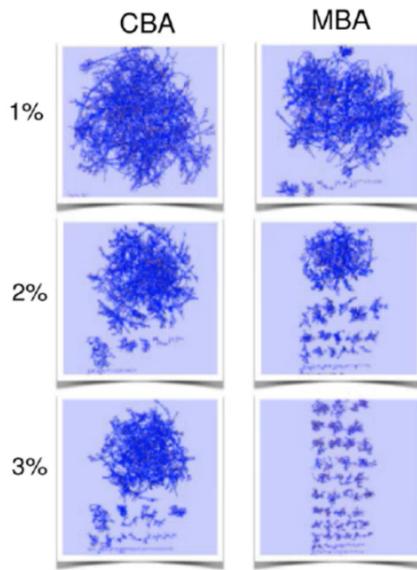


Questo è un set di curve di robustezza per attacchi su rete scale free. Le differenze tra attacchi random e al grado sono significative.

Si noti però come il Modularity Base Attacck (MBA) sia molto più efficiente dell'attacco al grado o alla betweenness.

❓ Perchè?

## Effect



Nelle figure di sinistra vediamo cosa succede quando si effettua un attacco centrality based, in particolare al grado, ovvero attacca gli hub e i singoli nodi e genera un fenomeno per cui si mantiene il giant component. Il modularity based attacck invece all'inizio è estremamente efficiente in quanto **cerca le community e le isola**. Questo attacco è molto efficace sulle reti con una forte struttura a comunità.

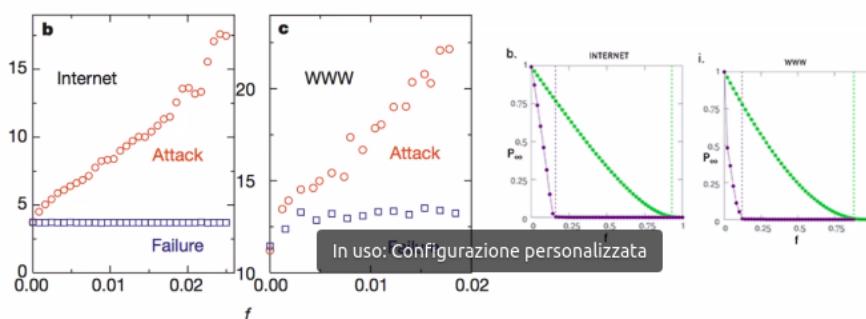
Questi strumenti sono molto potenti per analizzare non solo gli attacchi ma anche per analizzare un sistema a larga scala. La forza di questi metodi è l'astrazione avendo mappato la rete reale e fisica in

nodi ed edge.

La stessa proprietà che viene fuori dalla complex network analysis ad esempio che sia assortativa o no, va calata nel contesto della rete che si sta analizzando.

### "The Achilles' Heel of the Internet"

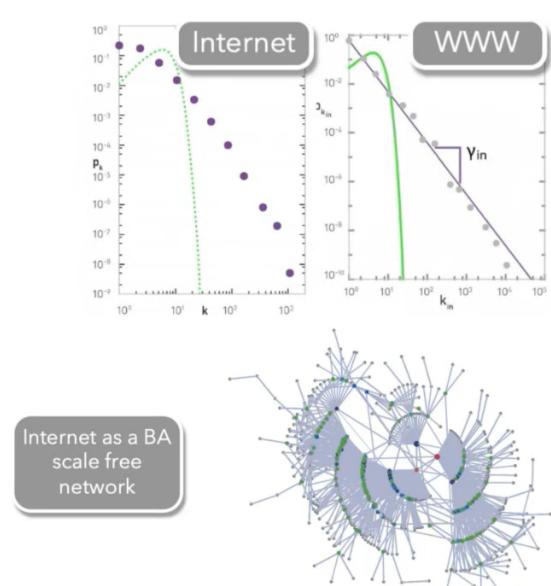
- ❑ Internet (and WWW) are very tolerant to random failures
  - ❑ Failures compromise nodes at random
- ❑ But they are extremely vulnerable to targeted attacks
  - ❑ Attacks towards high-degree nodes



Nel 2000, sulla copertina Nature venne pubblicato un articolo di Barabasi sul tallone di achille di Internet. Barabasi e il suo gruppo, si sono presi dei dati sulla connettività di Internet, hanno provato delle simulazioni di attacco trovando che sono reti scale free in quanto hanno la distribuzione del grado tipica delle reti power law, e hanno dimostrato che se attacco gli hub della rete internet posso distruggere la rete con pochissimo sforzo.

### Why?

- ❑ Internet (and WWW) shows very clearly a power-law degree distribution
- ❑ Therefore it is assumed they are scale-free networks
  - ❑ And can be described through the Barabasi-Albert model

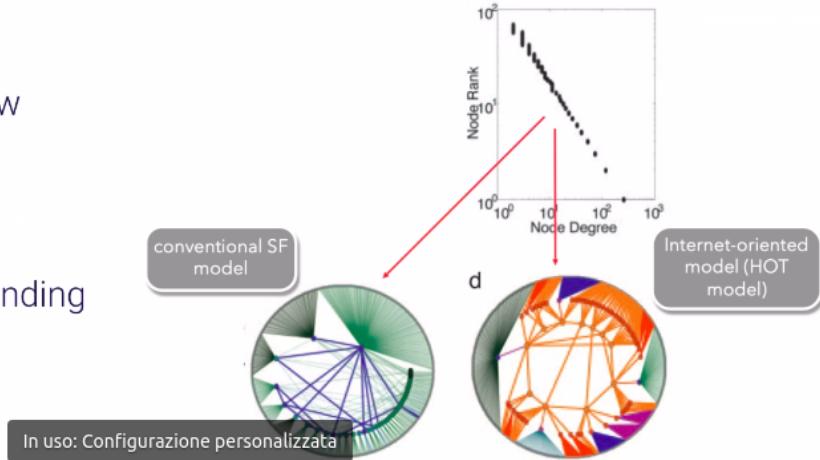


L'idea è che dall'analisi della degree distribution Internet e il web hanno una distribuzione del grado che si approssima bene con una power law dunque sono state modellate con una scale free. In particolare, dai dati hanno visto una rete power law e hanno riprodotto una rete scale free equivalente con lo stesso  $\gamma$ . Il modello usato però è stato quello di Barabasi-Albert. Hanno generato una rete Barabasi-

Albert equivalente e studiato la robustezza sul modello equivalente di Internet.

## Is BA the only possible model for SF networks?

- From a given power-law degree distribution there are many ways to generate a corresponding network topology

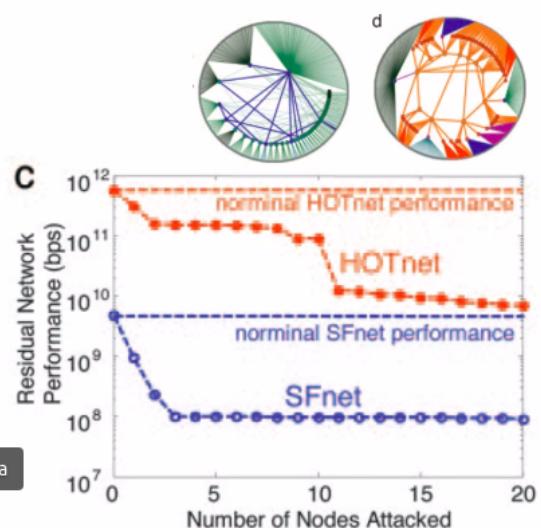


Poco dopo questi risultati pubblicati, la comunità di chi Internet l'ha veramente costruita, si è chiesta se il modello di Barabasi rispecchiava la realtà di Internet. In particolare Barabasi ha preso **UNO** dei possibili modelli di rete senza chiedersi se il modello rispondeva ai principi costruttivi di Internet. Nel plot vediamo una degree distribution empirica, dopo di che faccio il fitting e nei due pallini a sinistra ho una BA equivalente, a destra vediamo invece una rete con la stessa degree distribution ma con il modello HOT tenendo a mente come Internet davvero è costruita. La BA ha degli HUB al **centro** della rete dunque gli hub sono effettivamente coloro i quali collegano i nodi alla rete mentre nel modello di destra gli HUB tendono a essere **periferici** ovvero a non stare al centro della rete, non vuol dire però che questi non siano HUB. L'aspetto fondamentale però è che l'HUB non collega direttamente i nodi periferici al resto del mondo ma li collega a un nucleo interno di pochi nodi che hanno un grado basso ma che sono fortemente collegati tra di loro dando connettività planetaria.

Gli HUB di internet sono normalmente costituiti da un nucleo di nodi molto corazzati e con collegamenti molto capienti che collegano in forma a mesh i diversi operatori telefonici tra di loro.

## Consequences on Robustness

- ❑ Residual capacity after removal of links in the two models
  - ❑ Sequential targeted attack towards the "worst-case" node
    - ❑ worst-case for SF: highest-degree node
    - ❑ worst-case for HOT: highest-capacity node
- ❑ In general, the capacity of HOT is much higher than SF
- ❑ HOT is very resilient to worst-case attacks
  - ❑ as the core is well-connected
- ❑ HOT is also resilient to degree-based attacks
  - ❑ as high-degree nodes are at the periphery



In alto vediamo la curva di attacco al grado rispetto al modello HOT in basso invece rispetto al modello BA. E' vero che da un punto di vista teorico ci sono gli HUB e internet ha un tallone di achille, ma se attacco Internet rispetto al grado di certo faccio perdere connettività ad alcuni clienti di quell'HUB ma internet continua a funzionare.

## Complex Networks and Domain Knowledge

- ❑ Complex networks techniques are extremely interesting and powerful
  - ❑ Describe the macroscopic features of networks with simple and compact indices
  - ❑ Observation of such indices can tell much about the network properties and its behaviour
- ❑ But, **domain knowledge** is always necessary
  - ❑ Blindly trusting macroscopic features (degree distribution, correlation coefficient, ...) without checking whether simple models correspond to real networks can lead to bold oversimplifications!