Large Graphs Mining

Theory and Applications

Cédric Gouy-Pailler

cedric.gouy-pailler@cea.fr

Organisation du cours (1/2)

- 28/11/2018 :
  - Introduction
  - Bases de théorie des graphes
  - Statistiques globales et analyse des liens
- 03/12/2018 :
  - Clustering
  - Détection de communautés
- 12/12/2018 :
  - Séance informatique sur la détection de communautés/clustering
  - TP rendu en fin de séance (3/8 de la note du cours)

Organisation du cours (2/2)

- 09/01/2019 :
  - Graph embeddings
  - Graph Neural Networks
- 16/01/2019 :
  - Séance informatique sur le cours du 09/01/2019
  - Rendu en fin de séance (3/8 de la note finale)

- Dernier quart de la note : présence (5 points assurés par la présence)

Networks and complex systems

- Complex systems around us
  - **Society** is a collection of 6 billions individuals [social networks]
  - **Communication systems** link electronic devices [IoT -- shodan]
  - **Information and knowledge** are linked [Wikipedia, freebase, knowledge graph]
  - Interaction between thousands of **genes** regulate life [proteomics]
  - **Brain** is organized as a networks of billions of interacting entities [neurons, neuroglia]

What do these networks have in common and how do we represent them?

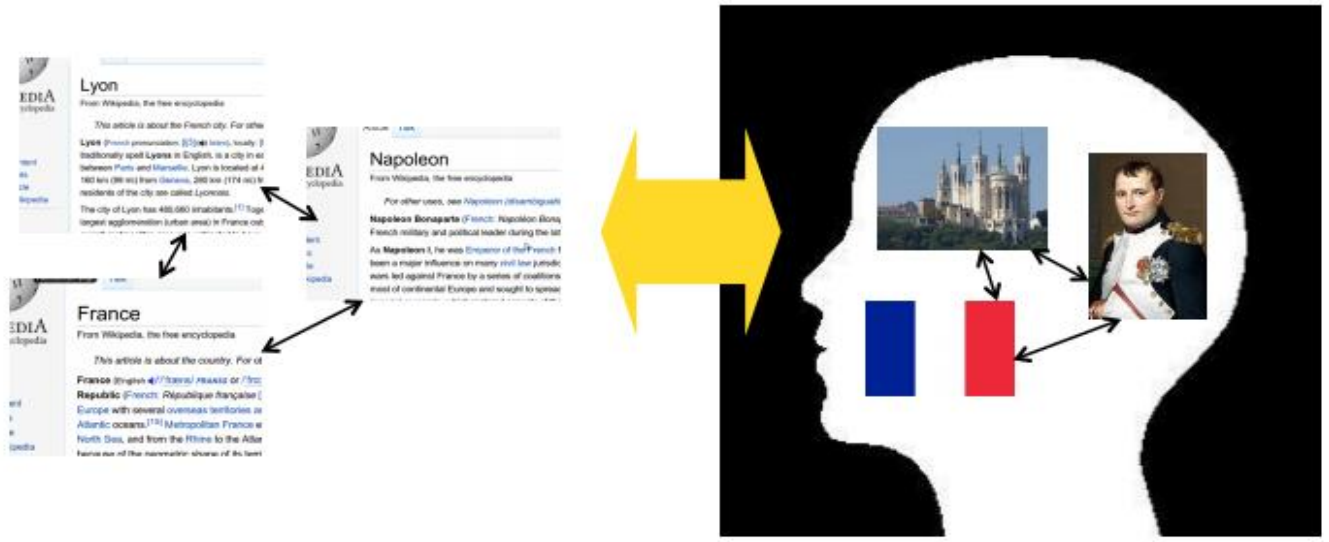
Examples from M2M communication systems

- IoT search engine
  - Shodan is the world's first search engine for Internet-connected devices
  - 1.5 billion interconnected devices
  - “The Terrifying Search Engine That Finds Internet-Connected Cameras, Traffic Lights, Medical Devices, Baby Monitors And Power Plants” [forbes, 2013]

Example: Information and knowledge

- How do we build maps of concepts?
  - Wikipedia
  - Freebase (57 million topics, 3 billion facts)

Wikipedia size & users	
English articles:	5,022,156
Total wiki pages:	37,920,487
Average revisions:	21.16
Total admins:	1,330
Total users:	26,876,507
UTC time: 12:02 on 2015-Dec-2	



Understand how humans navigate Wikipedia

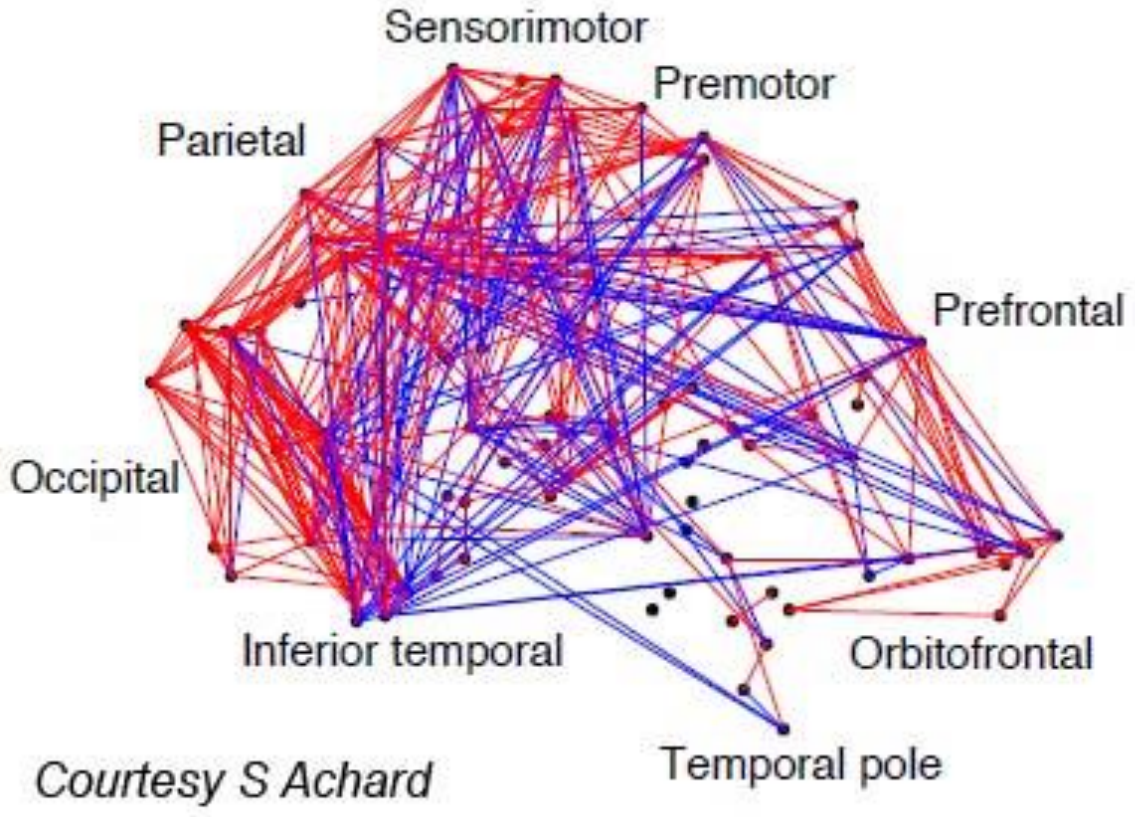
Get an idea of how people connect concepts

[West, Leskovec, 2012] get an idea of how people connect concept

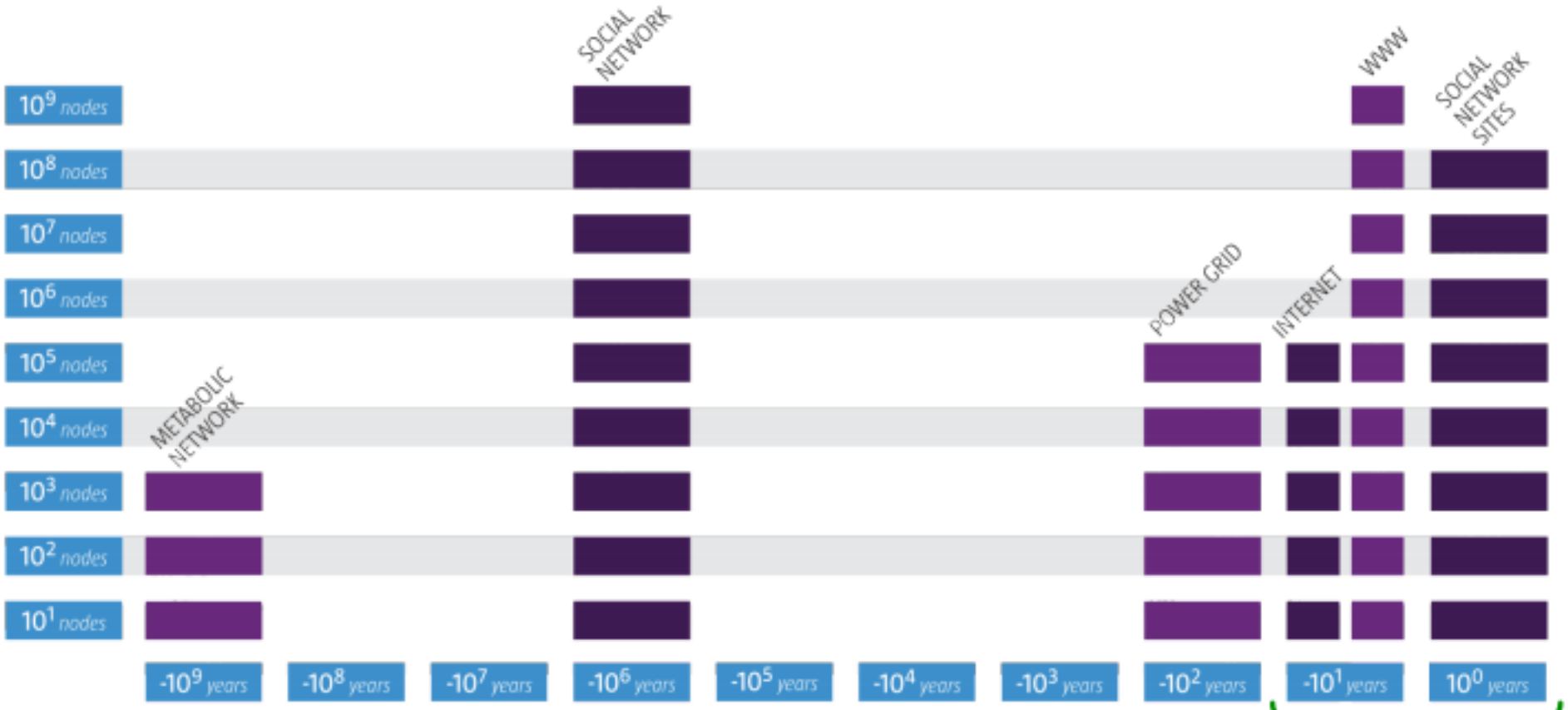
Examples from brain networks

- Brain network has between 10 and 100 billion neurons
- Connectivity networks:
  - Diffusion tensor imaging
  - Fiber tracking
  - Physical connections
- Functional connectivity
  - How electrical or BOLD activities are correlated (or linked)

- Understand brain lesions
- Epilepsy



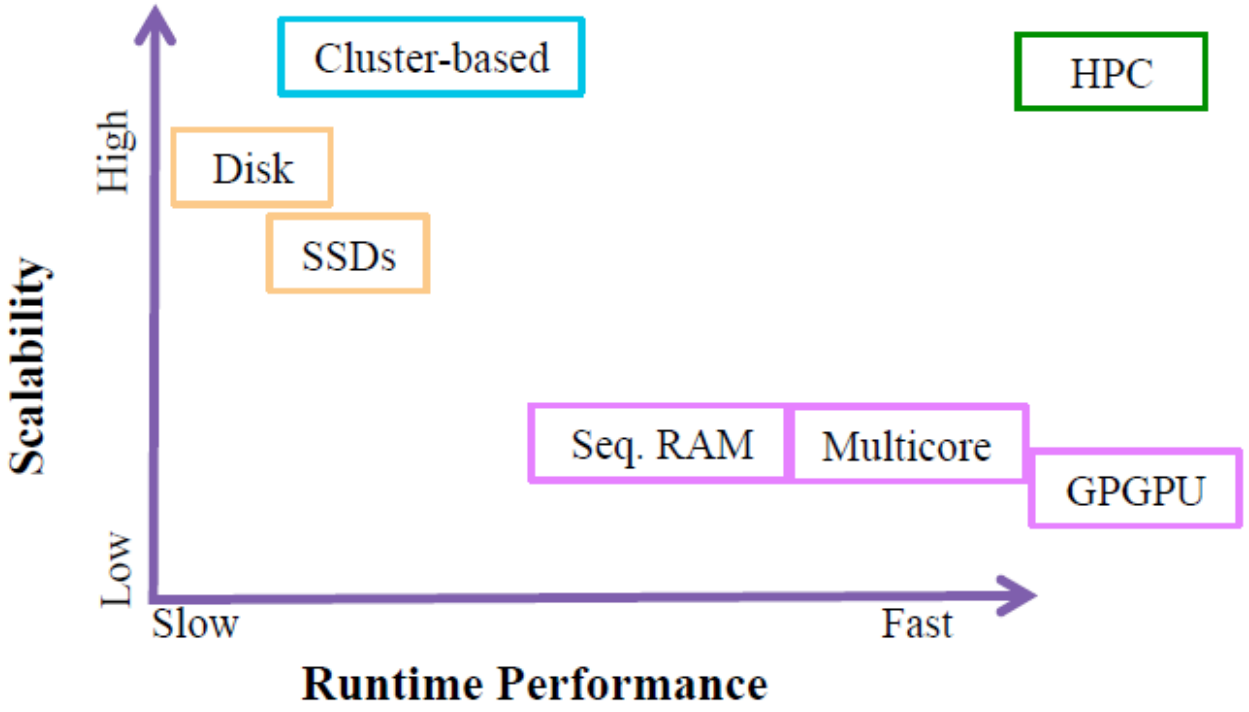
Why studying networks now?



Trade-offs in large graph processing

Representations, storage, systems and algorithms

Diversity of architecture for graph processing



- In-Memory
- On Disk
- Distributed
- Highly Customized SuperComputer

GPGPU: Gunrock

- GPGPUs offer thousands of parallel processing units,
  - Not many graph algorithms can leverage its fine-grained parallelism
  - Requires considerable expertise to write efficient code for processing graphs with GPGPUs
  - Library approach as opposed to systems approach
  - Ready-to-use graph algorithms already implemented on GPGPU (Gunrock from UC Davis)

- Scalability:
  - Subgraphs needs to fit in GPU memory which is much less than CPU memory
    - Moving the graph data between CPU and GPU memory slow

- Running time:
  - Unless embarassingly parallel algorithm, thousands of processing units typically speed-up the algorithm by a factor between one and hundred

Multicore: LIGRA, Galois

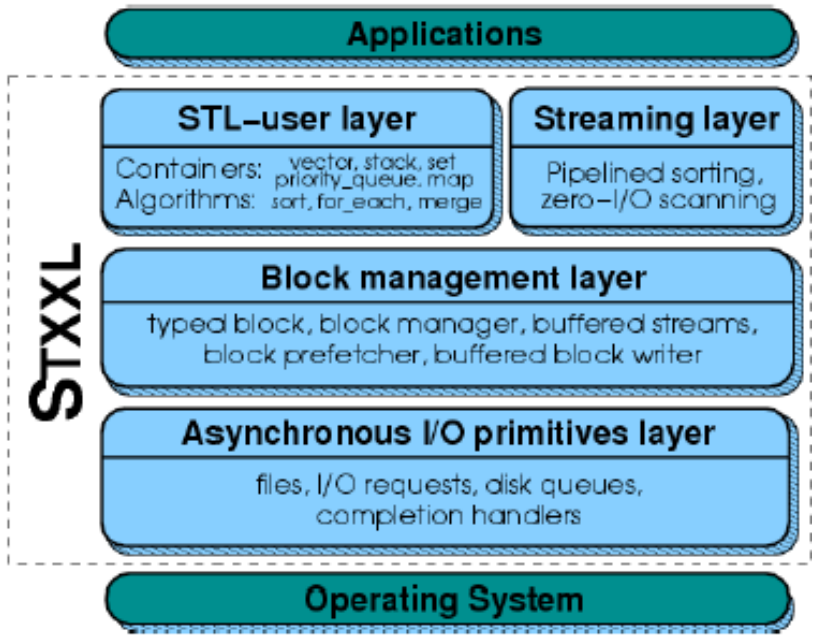
- Multi-cores with coarse-grained parallelism and shared memory are better suited to parallelize graph algorithms
  - Graphs are irregular and graph algorithms have a lot of branching instructions and random accesses
  - Systems such as Ligra and Galois define low-level primitives to support graph algorithms for multicores

- Scalability:
  - Graph needs to fit in the main memory restricting the size of graphs

- Running time:
  - Results in very fast parallel implementations
  - Typical multi-core architectures have 2-64 cores, but the number is increasing

Disk: STXXL, GraphChi

- Large body of literature to design and implement I/O-efficient algorithms and data structures
  - Replace random accesses by structured accesses to alleviate I/O-bottlenecks
  - STXXL: Library for I/O-efficient data structures
  - GraphChi: System for quickly deploying vertex-centric algorithms I/O-efficiently



- Scalability:
  - Can process graphs as long as graph and its intermediate copies can fit on disk
- Running time:
  - Web-graph with many billions of edges can be processed in less than an hour on a PC

SSD: PrefEdge

- SSDs provide faster random reads and provide I/O parallelism
  - PrefEdge: A prefetcher for graph algorithms that prefetches requests to derive maximum throughput from SSDs
  - Write bandwidth starts matching read bandwidth, but write latencies are still significantly more than read latencies
  - Graph algorithms often involve updating the state of nodes/edges: Random write accesses
- Scalability:
  - SSDs with capacity in TBs readily available

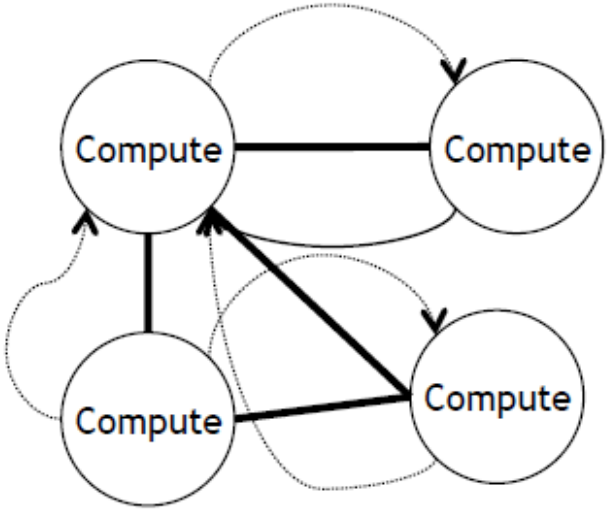
- Running time:
  - On a Twitter graph with around 1.6 billion edges, Dijkstra's SSSP algorithm with PrefEdge runs within a factor 5 of the in-memory, despite using only 15-20% of the main memory required
  - I/Os still remain the bottleneck

Distributed in-memory systems

- Restricted but intuitive computation model: bulk synchronous processing
  - "MapReduce for graphs", batch style
  - Typically vertex-centric, data exchanged along graph edges
  - Synchronous and asynchronous variants
- Scalability:
  - Designed to support massive scale for iterative processing
  - Replication for fault-tolerance
- Running time:
  - Good performance for restricted computation model
  - Inter-machine communication over network becomes bottleneck
    - Performance strongly influenced by graph partitioning

Pregel GraphLab

APACHE GIRAPH



Distributed in-memory persistence: trinity, graphX, Horton+

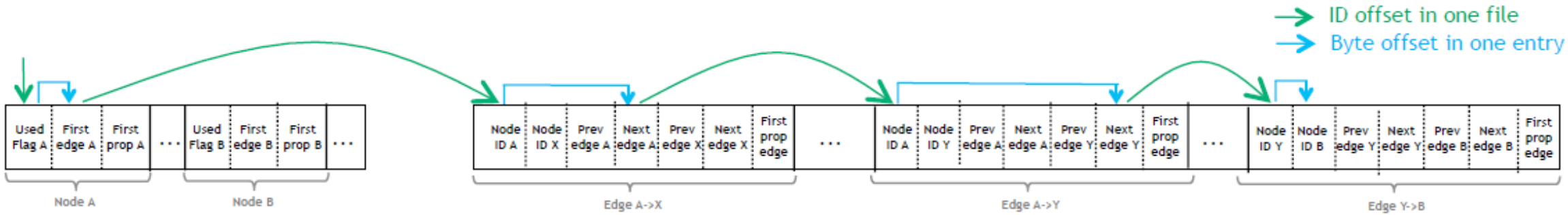
- Similar approach to distributed batch-style systems, but **keep** graph in memory
  - Supports interactive & online graph querying
  - Trinity: distributed "shared" memory cloud
  - GraphX: graph support on top of SPARK's resilient distributed data structures
  - Horton+, Green-Marl, Grace: in-memory graph systems (more or less) influenced by DBMS

- Scalability:
  - Same as distributed batch-style systems

- Running time:
  - Avoid reloading of graph from disk
  - Good performance also for database-like querying

Graph databases: neo4j, orientDB, titan

- Persistent storage solutions with native graph support or independent graph layer
  - Users think and query directly in graphs, no translation to tables or similar
  - Follow core principles from "traditional" DBMS, but end in different **trade-offs**
  - Strong support for persistency & one-shot queries, but typically limited in graph mining
- Scalability:
  - Often limited in distribution, some support NoSQL backends like Cassandra, HBase, etc.
- Running time:
  - Optimised for concurrency & (*local*) traversals that require several joins in an RDBMS



Node file: direct access via node ID

Edge file: direct access via edge ID, doubly linked list

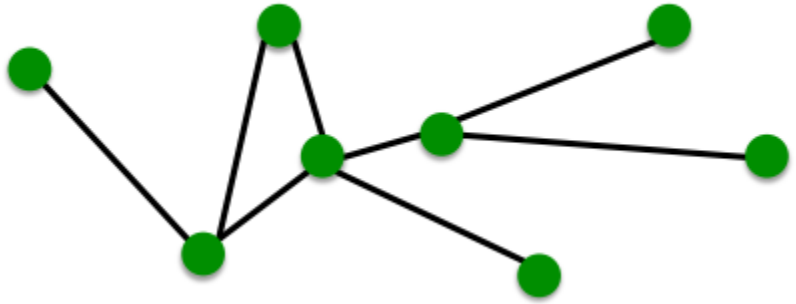
High-performance computing (HPC)

- HPC systems used for many scientific applications requiring fast and scalable graph processing
  - Specialized expensive hardware
  - Difficult to develop and optimize code, non-trivial maintenance of code-base and the system
  - Many implementations of BFS and SSSP on different supercomputers
    - BFS and SSSP on Cray MTA-2, Streaming graph analytics on Cray XMT
- Scalability:
  - HPC systems typically have very large distributed or "shared" memory
- Running time:
  - Graph500 Benchmark evaluates the suitability of different supercomputers for basic graph algorithms:
    - IBM BlueGene/Q with more than 1.5 million cores can traverse more than 23 trillion edges per second on a benchmark graph with more than 2 trillion vertices and 32 trillion edges

Notations and basic properties

Mathematical language

Notation and basic properties



- Objects: nodes, vertices N
- Interactions: links, edges E
- System: network, graph $G(N, E)$

Networks versus graphs

- Network often refers to real systems
  - Web, social network, metabolic network
  - **Language: network, node, link**

- Graph is mathematical representation of a network
  - Web graph, social graph (a Facebook term)
  - **Language: graph, vertex, edge**

Type of edges

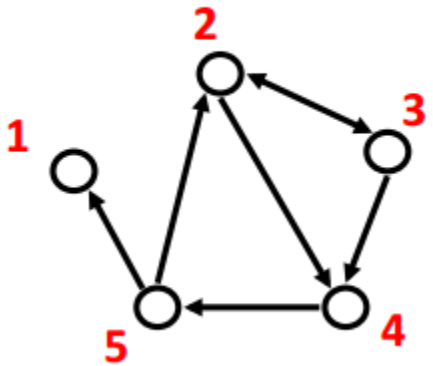
- Directed
  - A → B
    - A likes B, A follows B, A is B's child
- Undirected
  - A – B or A <--> B
    - A and B are friends, A and B are married, A and B are co-authors

Data representation

- Adjacency matrix
- Edge list
- Adjacency list

Adjacency matrix

- We represent edges as a matrix
  - $A_{ij} = \begin{matrix} 1 & \text{if node } i \text{ has an edge to node j} \\ 0 & \text{if node } i \text{ does not have an edge to node } j \end{matrix}$
  - $A_{ii} = 0$ unless the network has self-loops
  - $A_{ij} = A_{ji}$ if the network is undirected or i and j share a reciprocal edge



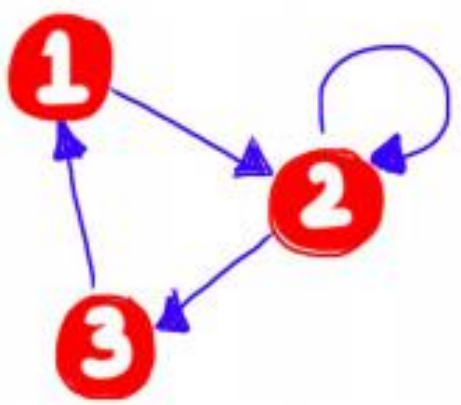
$$A = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 \end{pmatrix}$$

Adjacency matrix

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

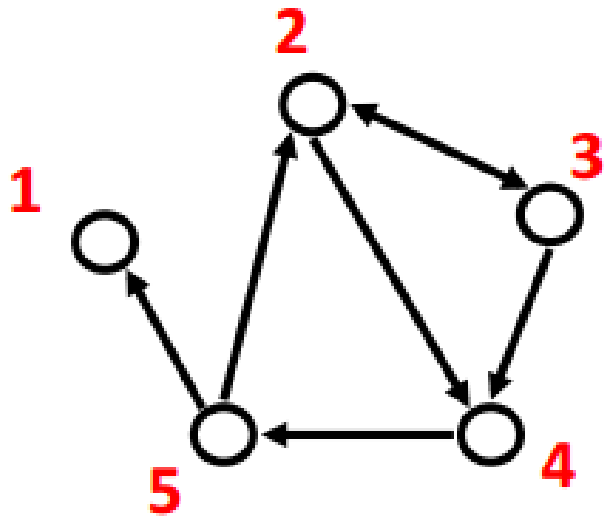
B)
$$\begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}$$

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



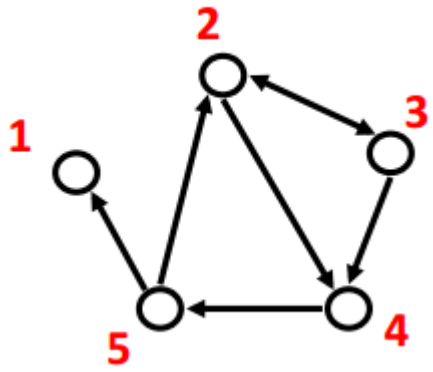
Edge list

- Edge list
  - 2 3
  - 2 4
  - 3 2
  - 3 4
  - 4 5
  - 5 1
  - 5 2



Adjacency list

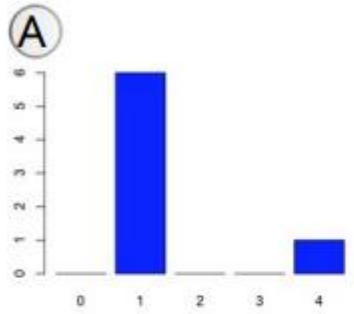
- Easier if network is
  - Large
  - Sparse
- Quickly access all neighbors of a node
  - 1 :
  - 2 : 3 4
  - 3 : 2 4
  - 4 : 5
  - 5 : 1 2

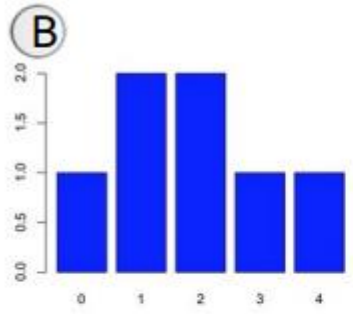


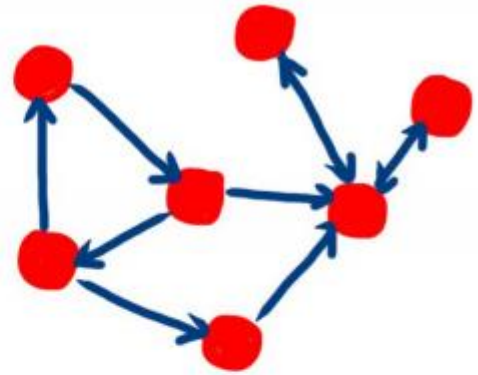
Degree, indegree, outdegree

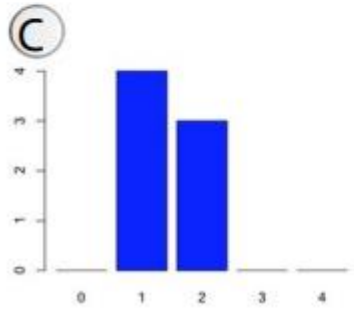
- Nodes properties
  - Local: from immediate connections
    - Indegree: how many directed edges are incident on a node
    - Outdegree: how many directed edges originate at a node
    - Degree: number of edges incident on a node
  - Global: from the entire graph
    - Centrality: betweenness, closeness
  - Degree distribution
    - Frequency count of the occurrence of each degree

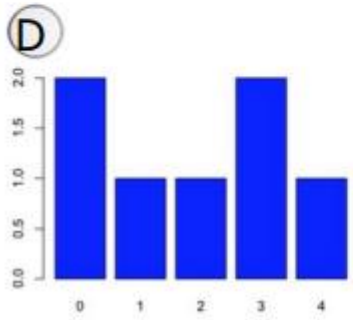
Guess the degree distribution





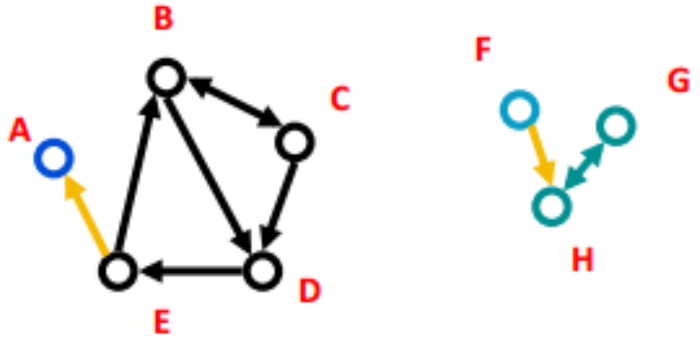






Connected components

- Strongly connected components
  - Each node within the component can be reached from every other node in the component by following directed links
  - B C D E
  - A
  - G H
  - F
- Weakly connected components
  - Weakly connected components: every node can be reached from every other node by following links in either direction
  - A B C D E
  - G H F
- In undirected graphs we just have the notion of connected components
- Giant component: the largest component encompasses a large portion of the graph



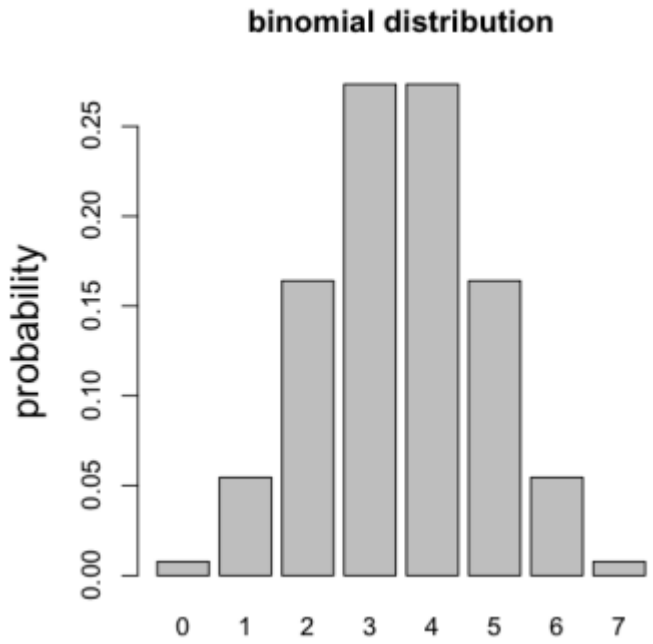
Classical tools for graph analysis

Random graphs, power law, and spectral analysis

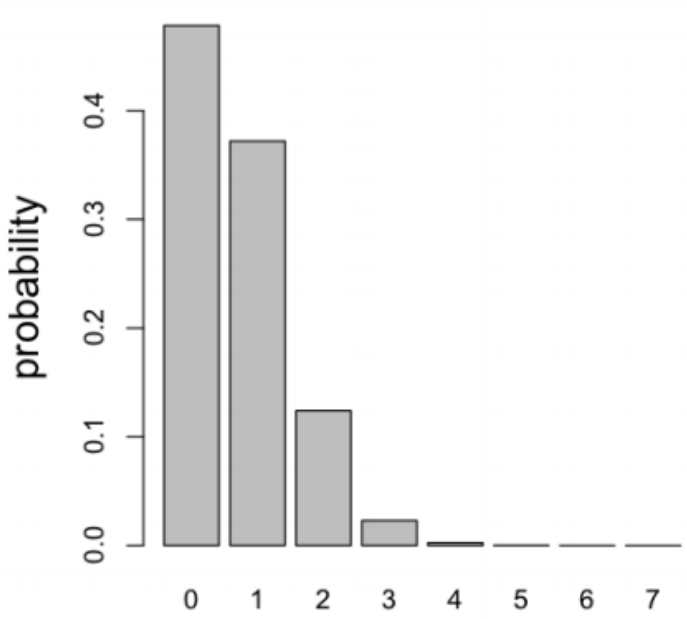
Erdos–Rényi (ER) random graph model

- Every possible edge occurs with probability $0 < p < 1$ (proposed by Gilbert, 1959).
- Network is undirected
- Many theoretical results obtained using this model
  - Average degree per node
    - $D_v \sim Binomial(n-1, p)$
    - $\mathbb{P}(D_v = k) = \binom{n-1}{k} . p^k . (1-p)^{n-1-k}$
    - $\mathbb{E}[D_v] = (n-1)p \approx np$

Erdos–Rényi (ER) random graph model



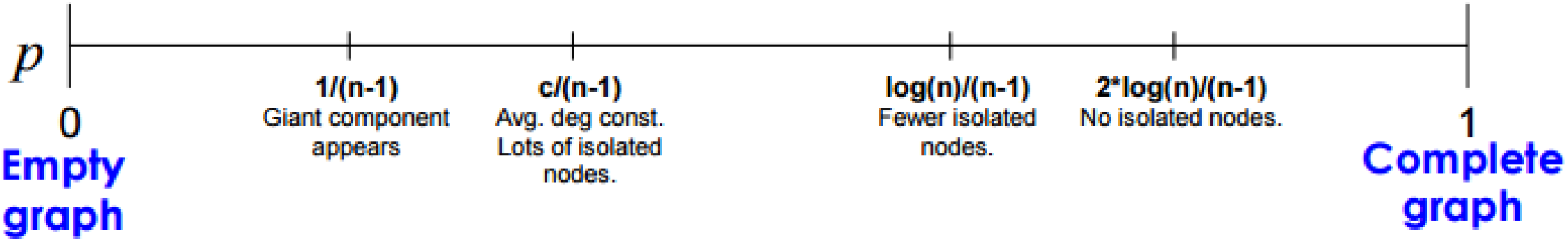
p=0.5



p=0.1

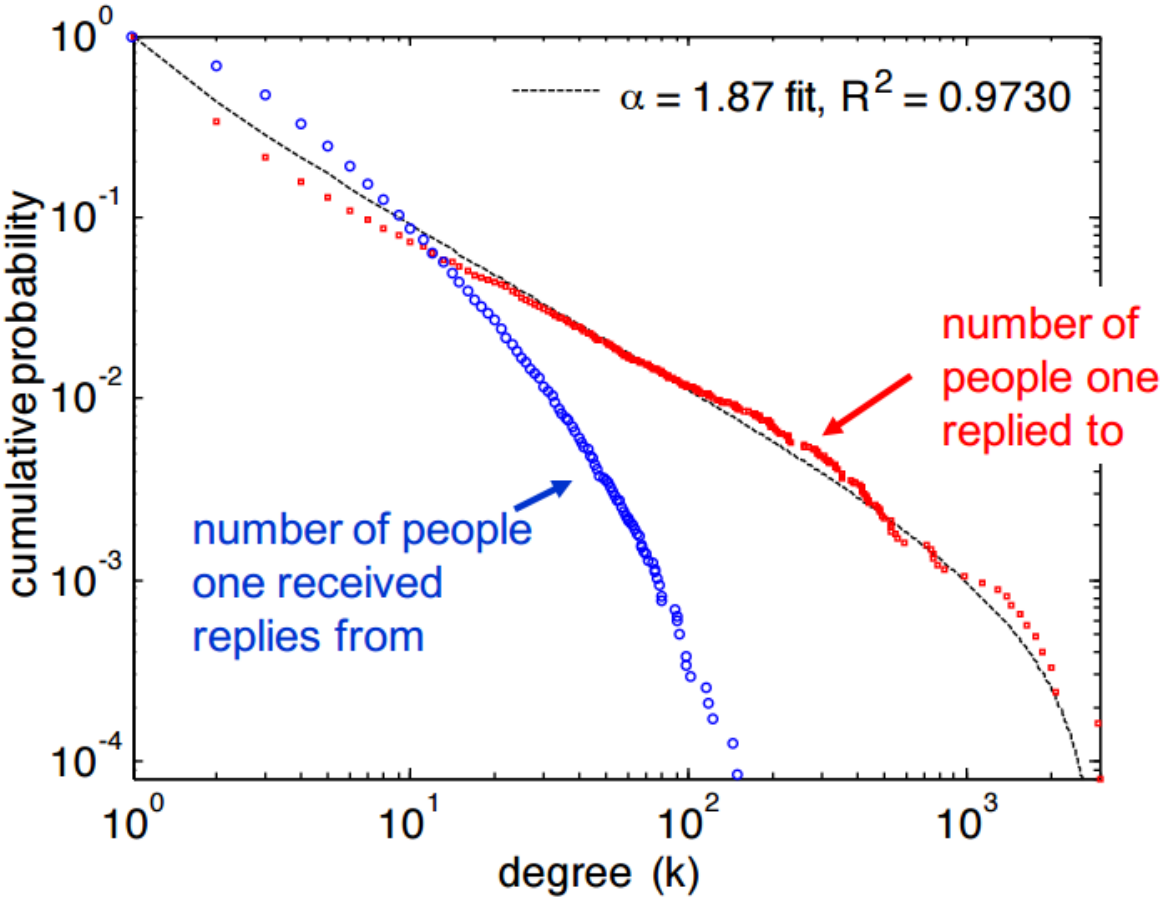
Not adapted to social networks organization

- Simple observation: no hub can appear
- Probability calculus describe appearance of isolated nodes and giant components as a function of p



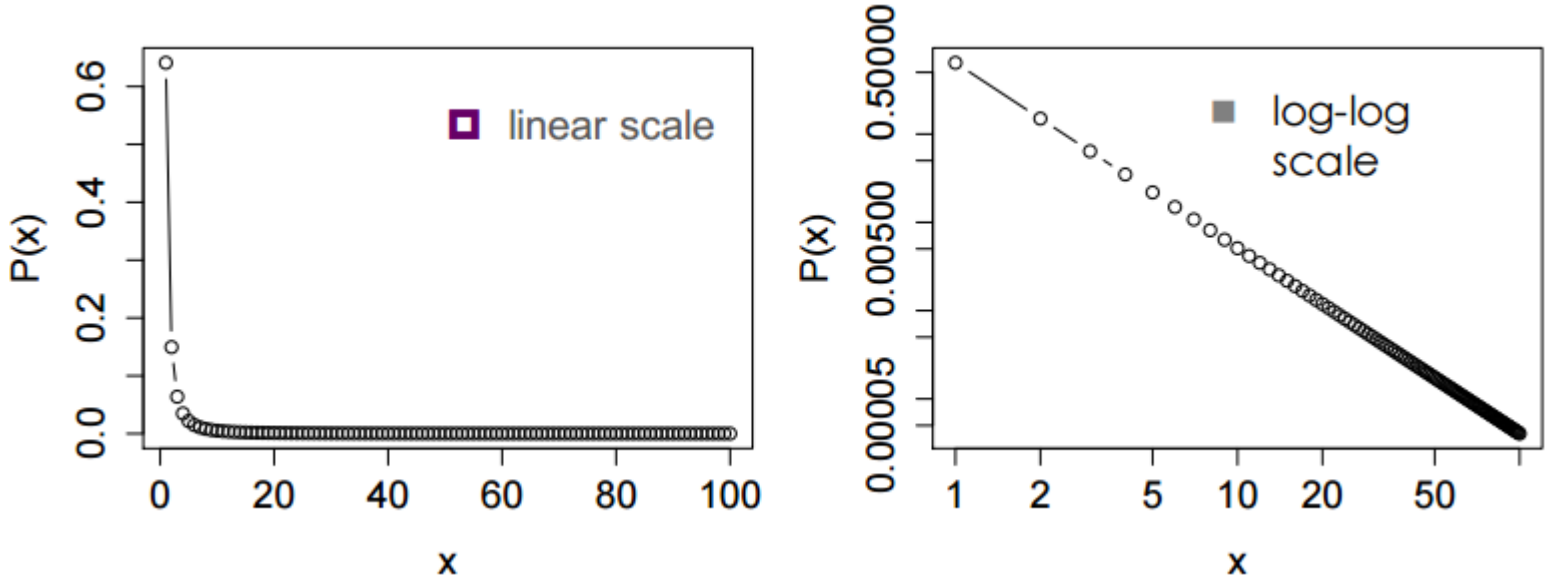
Power law graphs

- Online questions and answers forum



Power law distribution

- Distribution of degrees in linear and log-log scales
  - High skew (asymmetry)
  - Linear in log-log plot



Power law distribution

- Straight line on a log-log plot:

$$\log\big(p(k)\big) = c - \alpha \ln(k)$$

- Hence the form of the probability density function:

$$p(k) = C.k^{-\alpha}$$

  - α is the power law exponent of the graph
  - C is obtained through normalization

Where does “power law“ come from?

1. Nodes appear over time
   - Nodes appear one by one, each selecting m other nodes at random to connect to
   - Change in degree of node i at time t:

$$\frac{dk_i}{dt} = \frac{m}{t}$$

   - m new edges added at time t
   - The m edges are distributed among t nodes
   - Integrating over t:

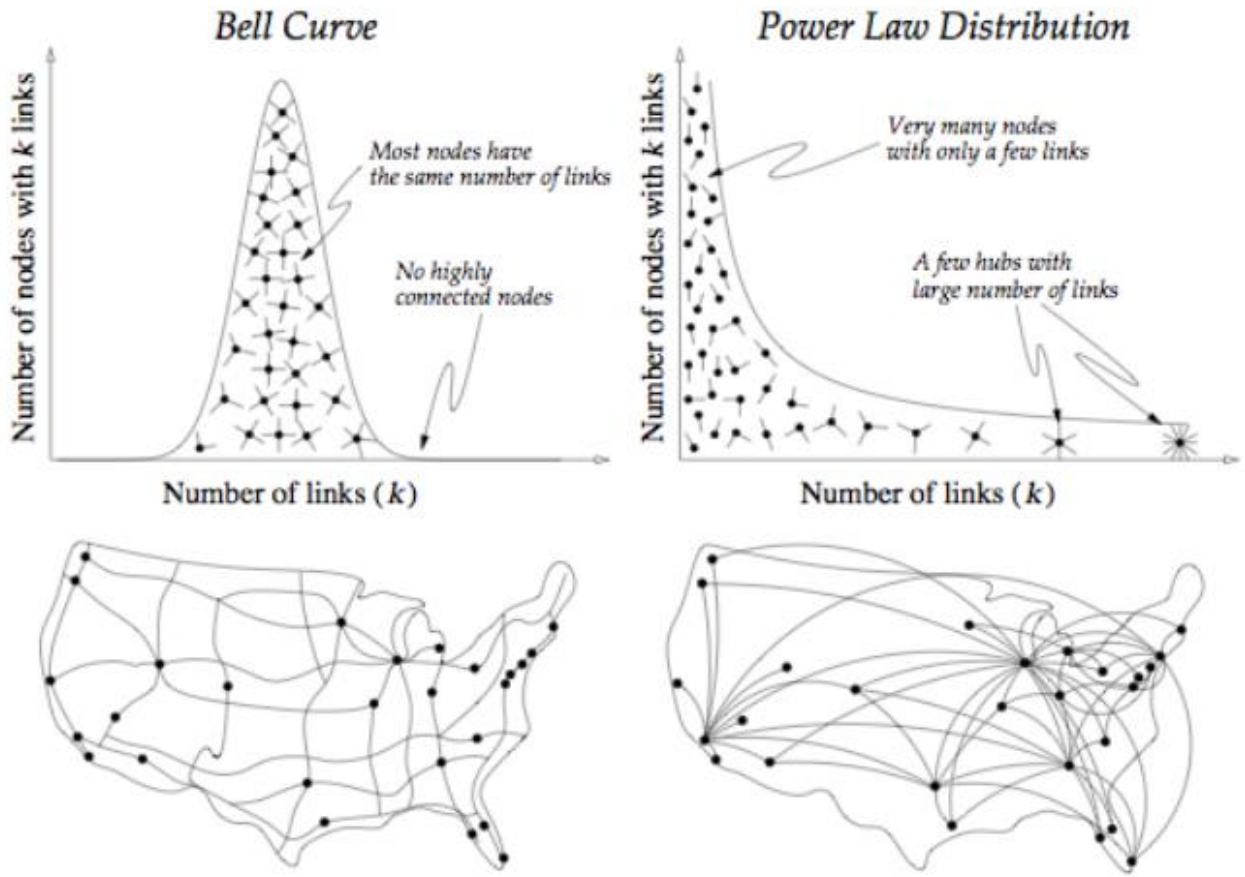
$$k_i(t) = m + m.\log(\frac{t}{i})$$

   - (born with m edges)
   - What’s the probability that a node has degree k or less?

Where does “power law“ come from?

2. Preferential attachment
   - new nodes prefer to attach to well-connected nodes over less-well connected nodes
     - Cumulative advantage
     - Rich-get-richer
     - Matthew effect
   - Example: citations network [Price 1965]
     - each new paper is generated with m citations (mean)
     - new papers cite previous papers with probability proportional to their indegree (citations)
     - what about papers without any citations?
       - each paper is considered to have a “default” citation
       - probability of citing a paper with degree k, proportional to $k + 1$
   - Power law with exponent $\alpha = 2 + \frac{1}{m}$

Exponential versus power law

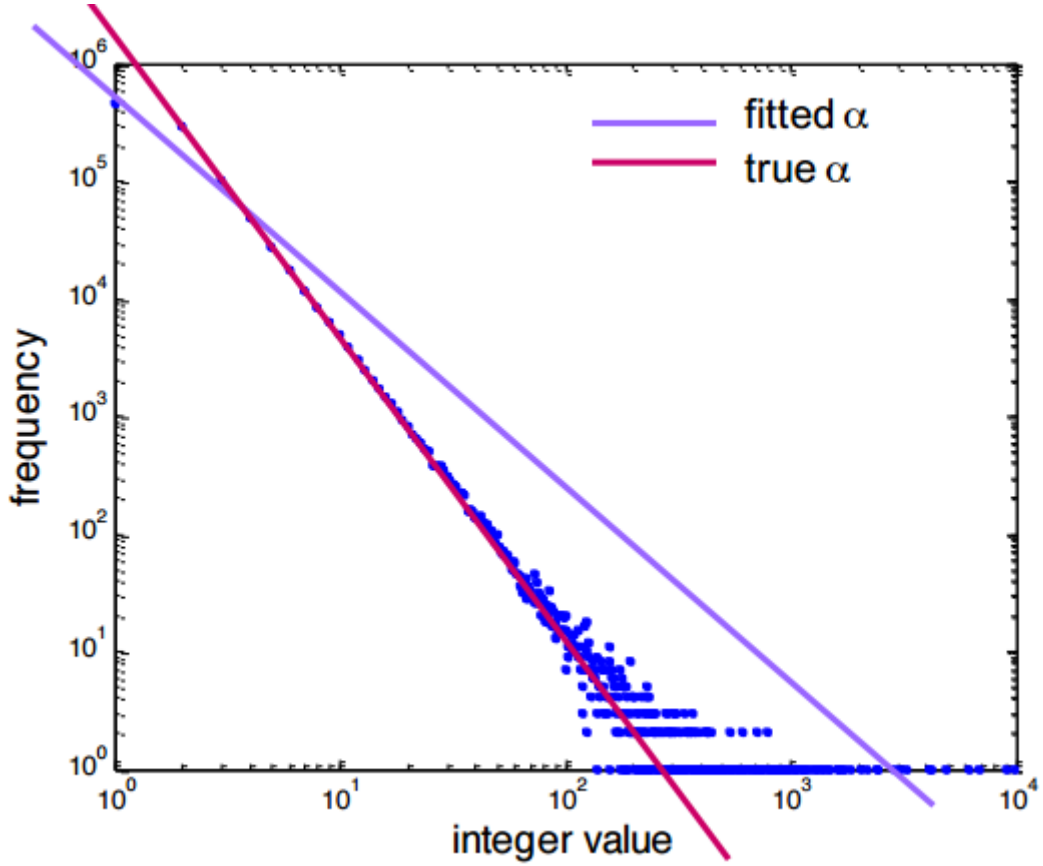


Distributions

Name	Distribution $p(x) = Cf(x)$ $f(x)$	C
Power law	$x^{-\alpha}$	$(\alpha-1)x_{\min}^{\alpha-1}$
Power law with cutoff	$x^{-\alpha}\mathrm{e}^{-\lambda x}$	$\frac{\lambda^{1-\alpha}}{\Gamma(1-\alpha,\lambda x_{\min})}$
Exponential	$\mathrm{e}^{-\lambda x}$	$\lambda \mathrm{e}^{\lambda x_{\min}}$
Stretched exponential	$x^{\beta-1}\mathrm{e}^{-\lambda x^{\beta}}$	$\beta\lambda \mathrm{e}^{\lambda x_{\min}^{\beta}}$
Log-normal	$\frac{1}{x}\exp\left[-\frac{(\ln x-\mu)^2}{2\sigma^2}\right]$	$\sqrt{\frac{2}{\pi\sigma^2}}\left[\operatorname{erfc}\left(\frac{\ln x_{\min}-\mu}{\sqrt{2}\sigma}\right)\right]^{-1}$

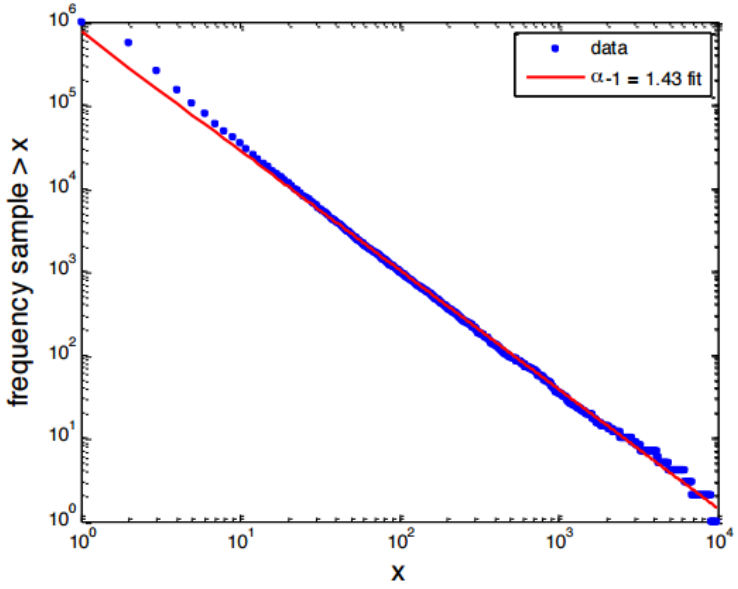
Fitting a power law distribution I

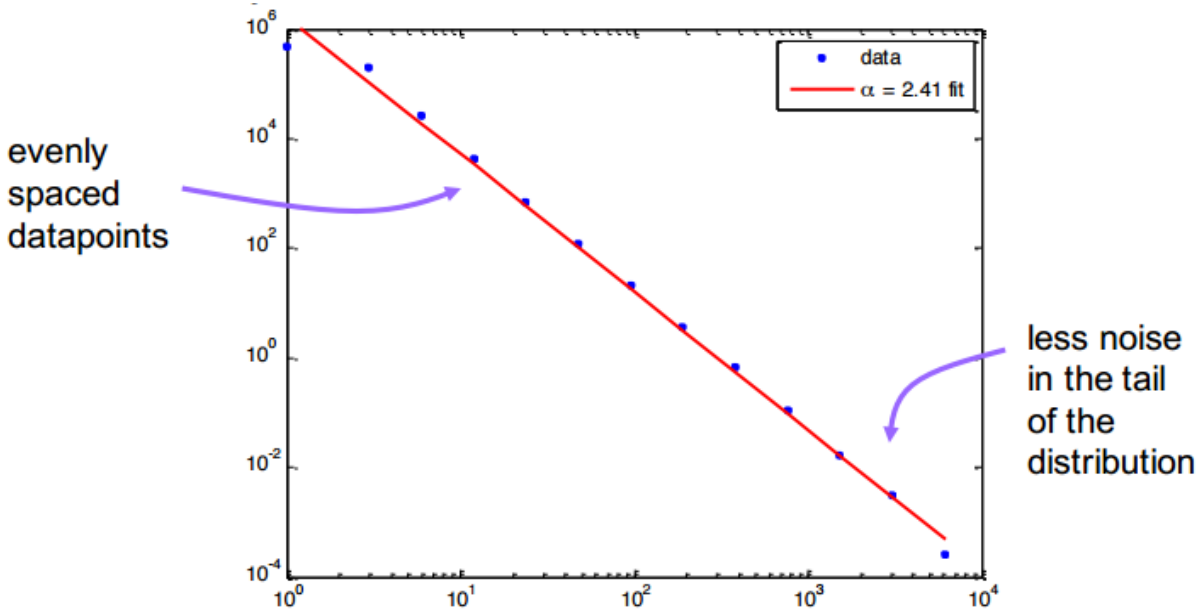
- Be careful about linear regression



Fitting a power law distribution II

- Approaches:
  - Logarithmic binning
  - Fitting with cumulative distribution
  - $\int c.x^{-\alpha} = \frac{c}{1-\alpha} x^{-(\alpha-1)}$





Small world graphs

- Watts-Strogatz, 1998
  - Alleviate properties of random graphs observed in reality
    - Local clustering and triadic closures
    - Formation of hubs
  - Algorithm
    - Given: number of nodes N, mean degree K, and a special parameter β, with $0 \leq \beta \leq 1$ and $N \gg K \gg \ln(N) \gg 1$.
    - Result: undirected graph with N nodes and $\frac{NK}{2}$ edges
  - Properties
    - Average path length → board definition
    - Clustering coefficient (global, local) → board definition
    - Degree distribution

Links analysis and ranking

Web data and the HITS and pagerank algorithms

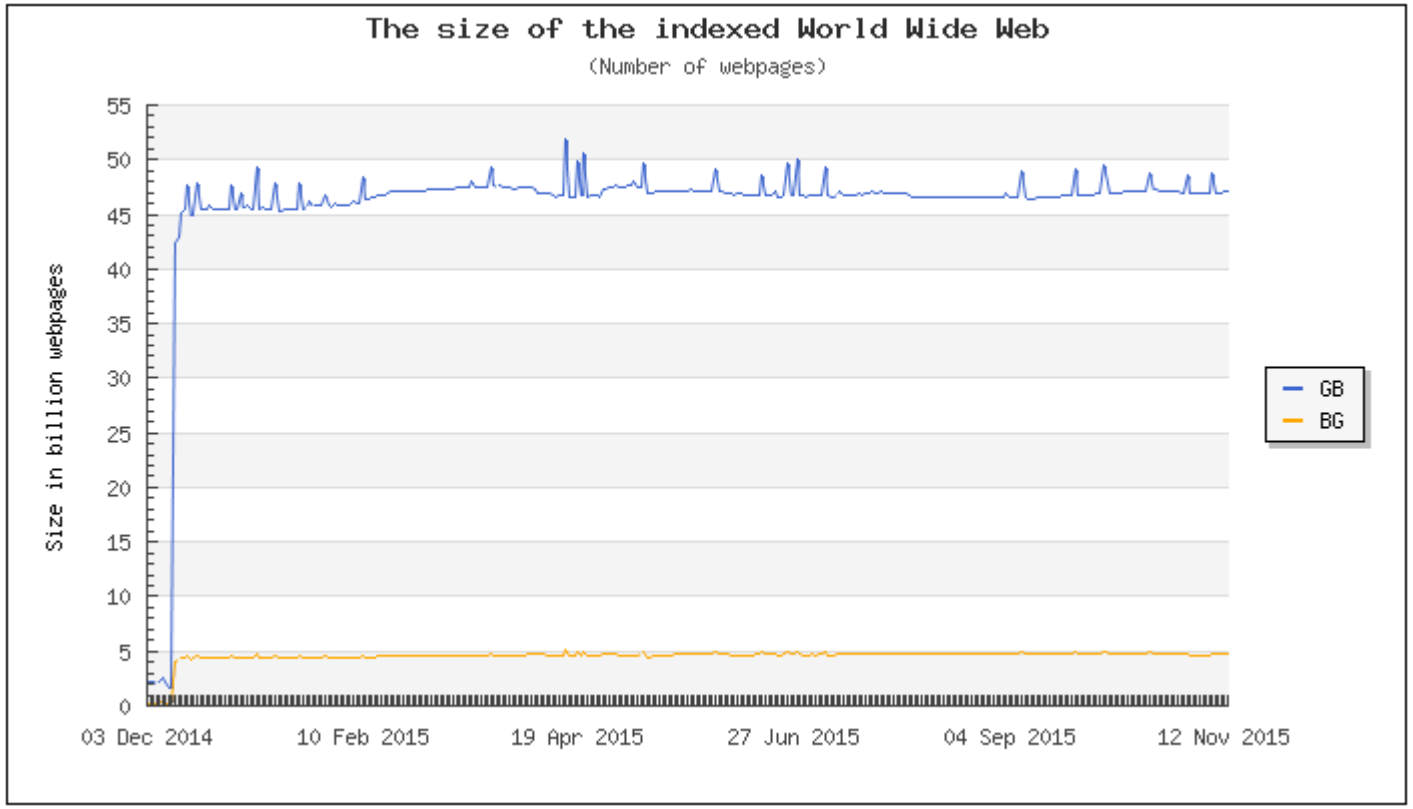
How do we organize the web?

- First simple solution:
  - Human curated
    - Old version of Yahoo for example
    - Web directories
  - Does not scale
  - Dynamics of the WWW
  - Subjective tasks

- Second solution
  - Web automated search
  - Information retrieval attempts to find relevant documents in a small and trusted set
    - Newspaper article, patents, scholar article, blog, forums, …
  - But the web is:
    - Huge
    - Full of untrusted documents
    - Random things
    - Web spam (false web pages)
  - We need good ways to rank webpages

Size of the indexed web

- The indexed web contains at least 4.73 billion pages (13 Novembre 2015)



Challenges of web search

- Web contains many sources of information
  - Who to trust?
  - Hint: trustworthy pages may point at each other!

- What is the best answer to query “newpapers”?
  - No single right answer
  - Hint: Pages that actually know about newspapers might all be pointing to many newspapers!

From web search to graph structures

- Web pages are pointing to each others, using hyperlinks.
  - This forms a networks in which:
    - Nodes are webpages themselves
    - Edges represent hyperlinks from some page point to another one (directed graph)
    -
- Web pages are not equally important.
- There is large diversity in the web-graph node connectivity.

Let’s rank the pages using the web graph link structure

Content of this part

- Two approaches to perform link analysis to compute importance of nodes in a graph
  - Hubs and authorities (HITS)
  - Page Rank

- Various notion of node centrality will be defined and experimented in practical sessions
  - Degree centrality: degree of node u
  - Betweenness centrality: #shortest paths passing through u
  - Closeness centrality: average length of shortest paths from u to all other nodes of the network
  - Eigenvector centrality: like pagerank

Hubs and authorities

HITS algorithm

History et basics

- Proposed by Jon Kleinberg in 1999
- Hubs
  - Some webpages serve as large directories used as compilations of a broad catalog of information that led users directly to authoritative pages.
  - Quality as an expert:
    - Total sum of votes of pages pointed to
- Authority
  - Webpage with high value content
  - Quality as a content:
    - Total sum of votes of expert

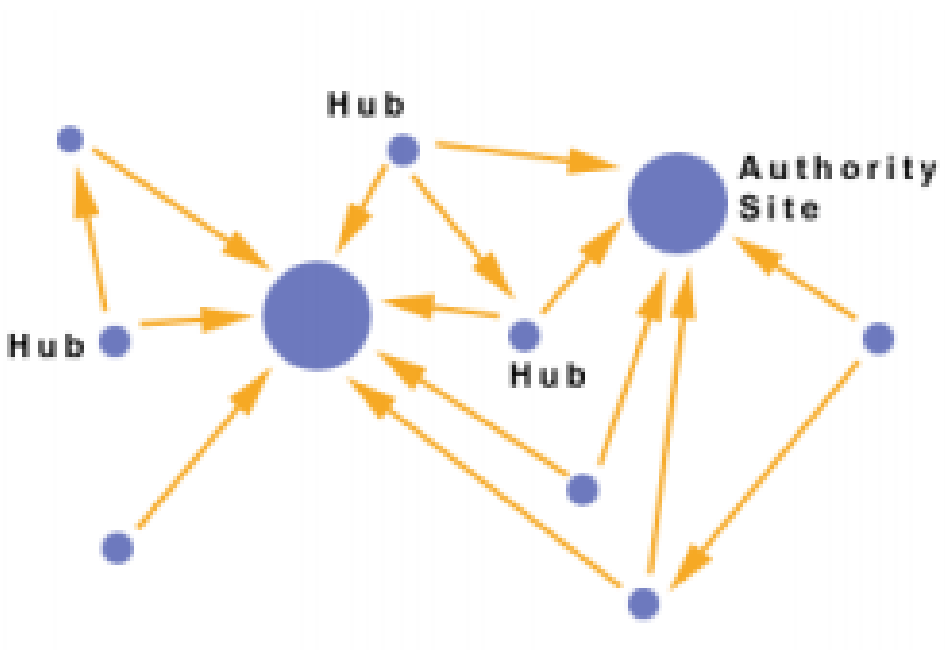
Hubs and authorities

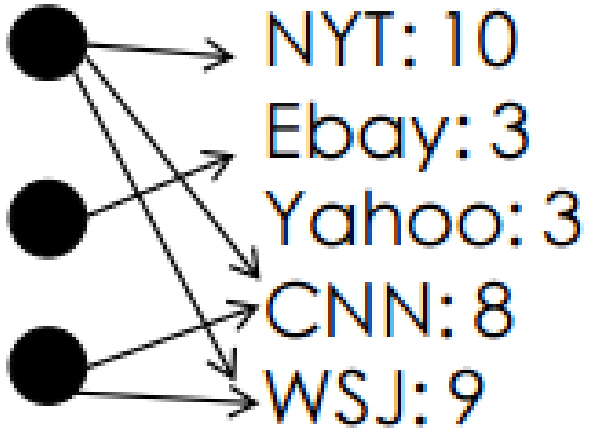
- Interesting pages fall into two classes

1. Authorities are pages containing useful information
   - Newspaper home pages
   - Course home pages
   - Home pages of auto manufacturers
2. Hubs are pages that link to authorities
   - List of newspapers
   - Course bulletin
   - List of U.S auto manufacturers

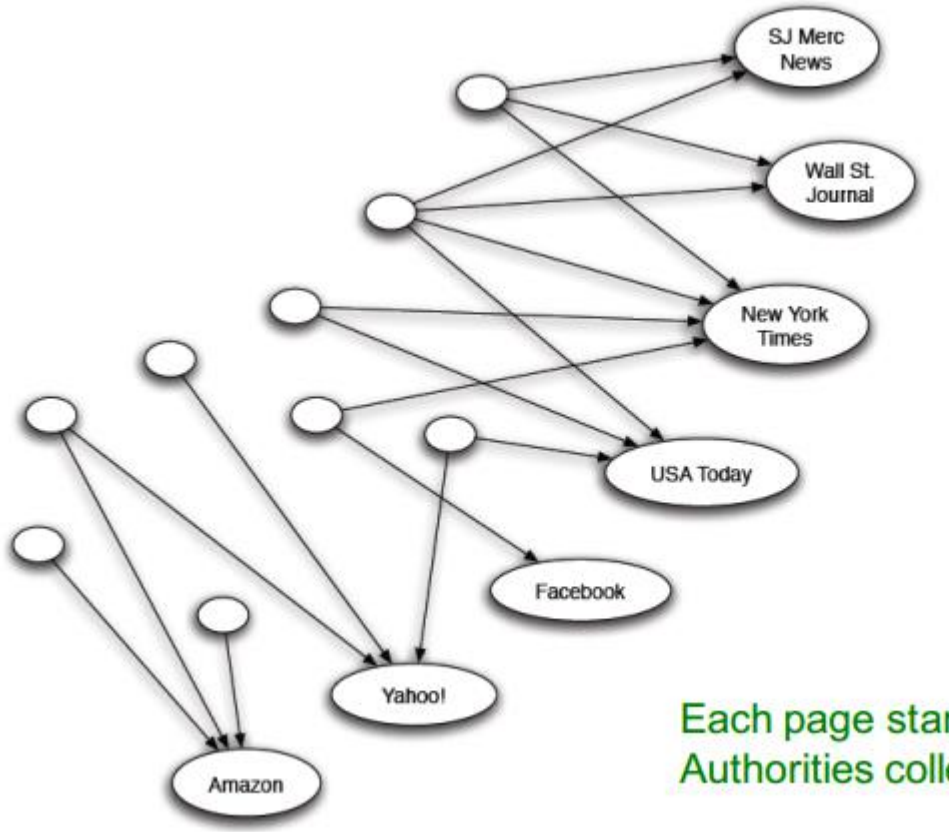
Each page has two scores → hub score and authority score

Illustration



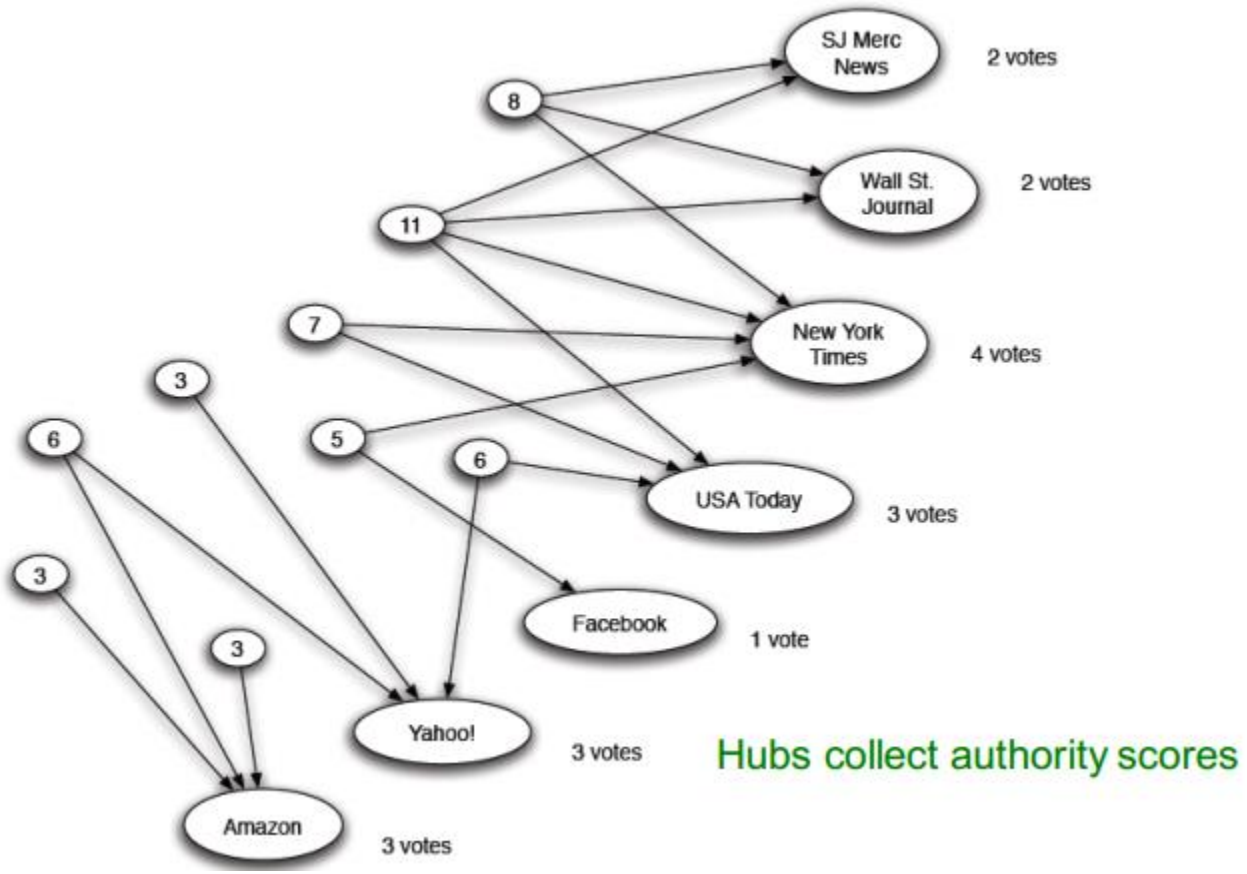


Authority

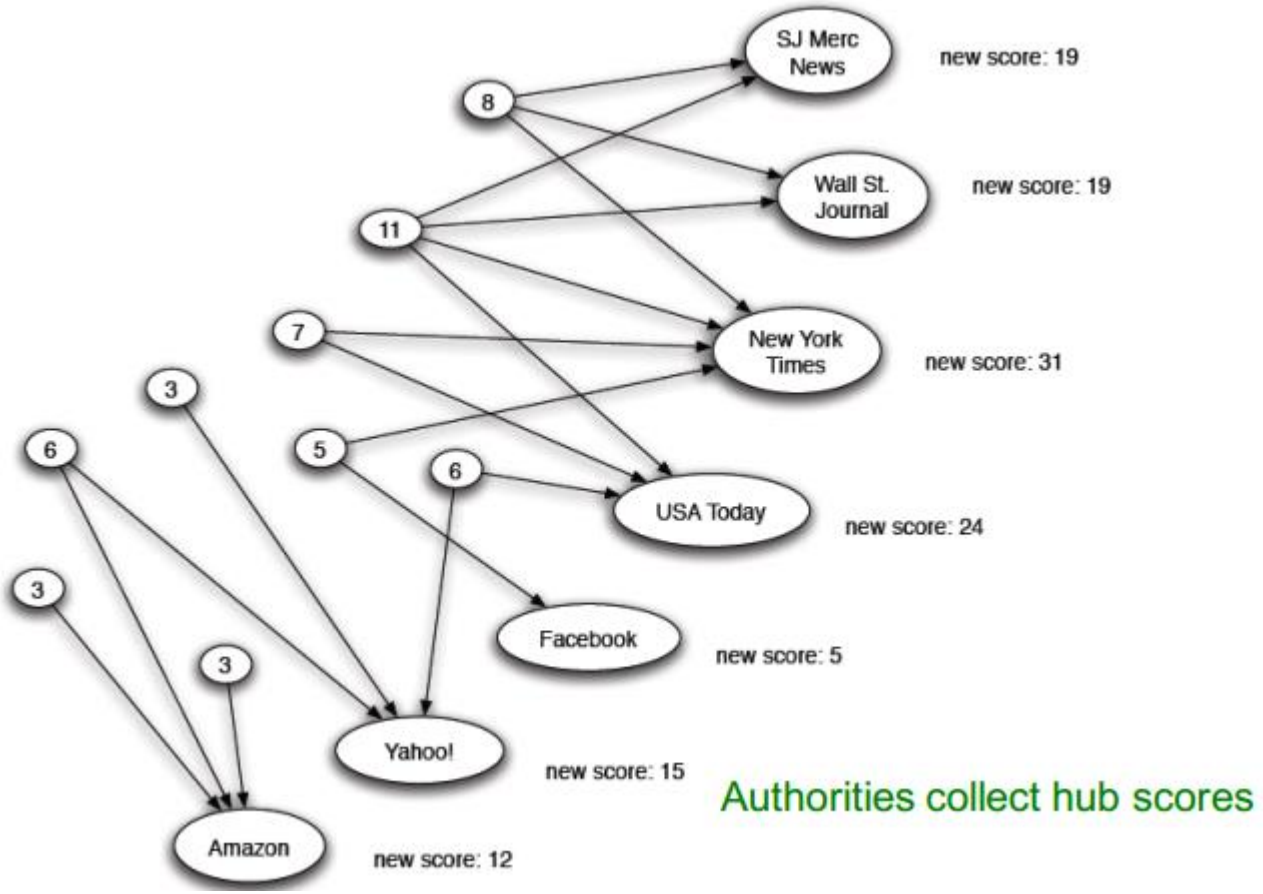


Each page starts with **hub score 1**
Authorities collect their votes

Hub


SJ Merc News
2 votes
8
Wall St. Journal
2 votes
11
New York Times
4 votes
7
3
6
5
6
USA Today
3 votes
3
Facebook
1 vote
3
Yahoo!
3 votes
Amazon
3 votes
Hubs collect authority scores

Reweighting



Mutually recursive definition

- A good hub links to many good authorities
- A good authority is linked from many good hubs

- Model uses two scores
  - Hub score and authority score
  - h and a vectors of scores, where the i-th element of the vector is the score of the i-th node.

HITS algorithm

- Each page i has two scores, a_i and h_i
- Initialize

$$a_j^{(0)} = \frac{1}{\sqrt{n}} \qquad h_j^{(0)} = \frac{1}{\sqrt{n}}$$

- Iterate until convergence
  - Authority: $\forall i \quad a_i^{(t+1)} = \sum_{j \to i} h_j^{(t)}$
  - Hub: $\forall i \quad h_i^{(t+1)} = \sum_{i \to j} a_j^{(t)}$
  - Normalize: $\sum_i (a_i^{(t+1)})^2 = 1 \qquad \sum_i (h_i^{(t+1)})^2 = 1$

HITS, vector notation

- Vectors $a = (a_0, a_1, \ldots, a_n)$ and h $= (h_0, h_1, \ldots, h_n)$
- Adjacency matrix $A \in \mathbb{R}^{n \times n}$: $A_{ij} = 1 \quad iif \; i \to j$
- Rewriting $h = Aa$ and $a = A^T h$
- Repeat until convergence
  - $h^{(t+1)} = Aa^{(t)}$
  - $a^{(t+1)} = A^T h^{(t)}$
  - Normalize $a^{(t+1)}$ and $h^{(t+1)}$

Power iterations

- We thus have:
  - $a^{(t+1)} = A^T A a^{(t)}$ and $h^{(t+1)} = A A^T h^{(t)}$
  - $A^T A$ is symmetric

- Eigenvectors and eigenvalues:
  - When HITS has converged, we reach a steady state
  - Authority a is eigenvector of $A^T A$ (associated with its largest eigenvalue)
  - Hub h is eigenvector of $A A^T$ (associated with its largest eigenvalue)

Convergence speed

- Theorem of Perron-Frobenius
- Details on board

In practice

1. First HITS identifies approximately 200 pages based on standard text-based approaches. Only pages containing words of the request can be selected at this step. Then every pages pointed by one of the first 200 pages is added to the subgraph G. The goal of this first step is to yield approximately 3000 pages.
2. The second step consists in finding among G the most relevant pages.

Pagerank

Pagerank algorithm

Links as votes

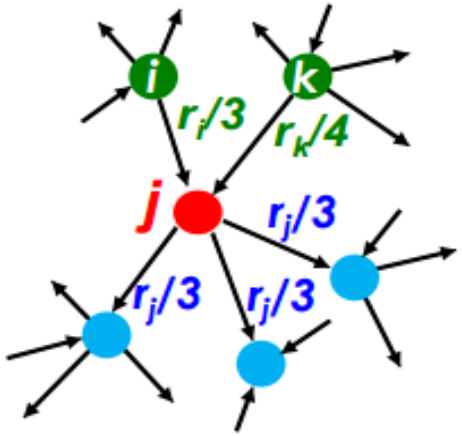
- A page is more important if it has more links (same idea as HITS)
  - Incoming? out-going?

- Think of in-links as votes
  - www.stanford.edu has 23400 in-links
  - www.joe-schmoe.com has 1 in-link

- Links are not equal
  - Links from important pages are more valuable
  - Recursive question!

Pagerank: flow model

- A vote from an important page is worth more
  - Each link's vote is proportional of its source page
  - Mathematically, if page i with importance r_i has d_i out-links, each link gets r_i/d_i votes.
  - Page j 's own importance r_j is the sum of the votes of its in-links.



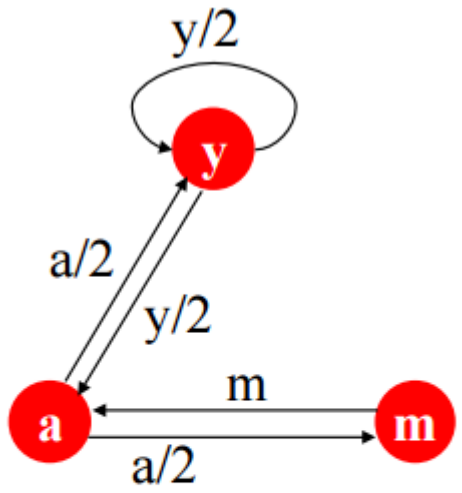
$r_j = r_i/3 + r_k/4$

Pagerank: flow model

- A page is important if it is pointed to by other important pages.
- Define a rank r_j for node j:

$$r_j = \sum_{i \to j} \frac{r_i}{d_i}$$

Out-degree of node i



"Flow" equations:

$r_y = r_y/2 + r_a/2$

$r_a = r_y/2 + r_m$

$r_m = r_a/2$

Matrix notation

- Stochastic adjacency matrix M
  - Let page j have d_j out-links
  - If $j \to i$ then $M_{ij} = \frac{1}{d_j}$
  - Columns sum to 1
- Rank vector r:
  - One entry per page
  - Normalized: $\sum_i r_i = 1$
- Equation flow can be written

$$r = Mr$$

$$(r_j = \sum_{i \to j} \frac{r_i}{d_i})$$

Random walk interpretation

- Imagine a random web surfer
  - At any time t, surfer is on some page i
  - At time $t + 1$, surfer follows an out-link from i uniformly at random
  - Ends up on some page j linked from i
  - Process repeats indefinitely
    - Let $p(t)$ vector whose i-th coordinate is the probability that the surfer is at page i at time t
    - So $p(t)$ is a probability distribution over pages

Random walk, stationary distribution

- Where is the surfer at time $t + 1$
  - Follow a link uniformly at random

$$p(t + 1) = M.p(t)$$

  - Suppose the surfer reaches a state

$$p(t + 1) = Mp(t) = p(t)$$

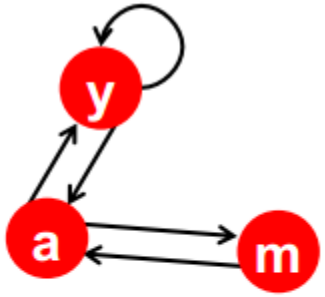
  Then $p(t)$ is the stationary distribution of the random walk.

  - Our original rank vector r satisfies $r = Mr$
    - So r is a stationary distribution for the random walk.

Power iteration method

- Power iteration
  - Set $r_j = \frac{1}{\mathrm{N}}$
  - 1: $\sum_{i \to j} \frac{r_i}{d_i} \to r'_j$
  - 2: $r' \to r$
  - If $|r - r'| > \varepsilon$ GOTO 1

$$\begin{pmatrix} r_y \\ r_a \\ r_m \end{pmatrix} = \begin{matrix} 1/3 \\ 1/3 \\ 1/3 \end{matrix}$$



	y	a	m
y	½	½	0
a	½	0	1
m	0	½	0

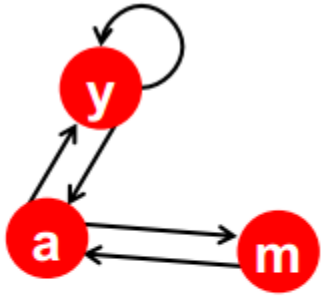
$r_y = r_y/2 + r_a/2$

$r_a = r_y/2 + r_m$

$r_m = r_a/2$

Power iteration method

- Power iteration
  - Set $\frac{1}{\mathrm{N} \to r_j}$
  - 1: $\sum_{i \to j} \frac{r_i}{d_i} \to r'_j$
  - 2: $r' \to r$
  - If $|r - r'| > \varepsilon$ GOTO 1

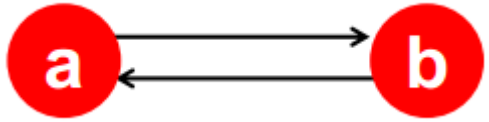


	y	a	m
y	½	½	0
a	½	0	1
m	0	½	0

$r_y = r_y/2 + r_a/2$
$r_a = r_y/2 + r_m$
$r_m = r_a/2$

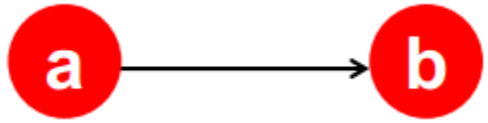
$$\begin{pmatrix} r_y \\ r_a \\ r_m \end{pmatrix} = \begin{matrix} 1/3 & 1/3 & 5/12 & 9/24 & & 6/15 \\ 1/3 & 3/6 & 1/3 & 11/24 & \ldots & 6/15 \\ 1/3 & 1/6 & 3/12 & 1/6 & & 3/15 \end{matrix}$$

Spider trap



Iteration:	0,	1,	2,	3…
r_a	1	0	1	0
r_b	0	1	0	1

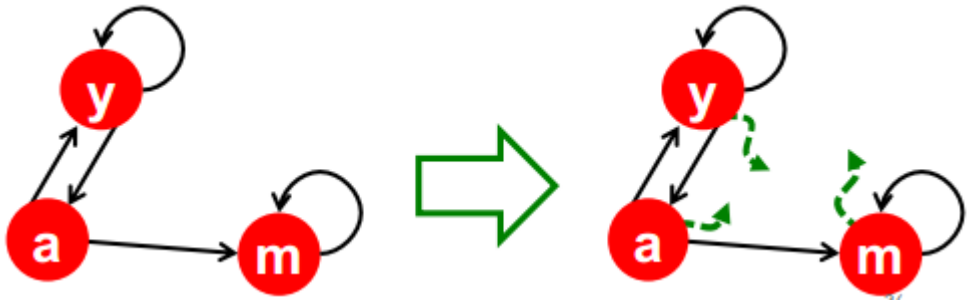
Dead end



Iteration:	0,	1,	2,	3…
r_a	1	0	0	0
r_b	0	1	0	0

Google solution: random teleports

- At each time steps, the random surfer has two solutions:
  - With probability β, follow a link at random
  - With probability $1 - \beta$, jump to a random page
  - Common values of β are in the range 0.8-0.9



Final pagerank equation

- [Brin, Page, 1998]

$$r_j = \sum_{i \to j} \beta \frac{r_i}{d_i} + (1 - \beta) \frac{1}{n}$$

Pagerank algorithm

- Let's define

$$M'_{ij} = \beta M_{ij} + (1 - \beta)\frac{1}{n}$$

- And thus we get what we want

$$r = M'r$$

- Note that M' is never materialized (sparse → dense)
- r is the stationary distribution of the random walk with teleports

Computational considerations

- Is it feasible to store vectors and matrix for the whole web?
  - Use 1 billion web page
- The sparse approach
  - Rearrange equation such that $r = \beta M r + \left[\frac{1-\beta}{N} \quad \ldots \quad \frac{1-\beta}{N}\right]^T$
  - Does it fit in RAM?

Web spamming

- Boosting approaches
  - Artificially increase relevance (or rank) of a web page
    - Spam with terms: manipulate text of pages to make the pages relevant for certain requests
    - Spam with links: build a linkage structure to artificially increase the pagerank score of a page

Web spamming: terms

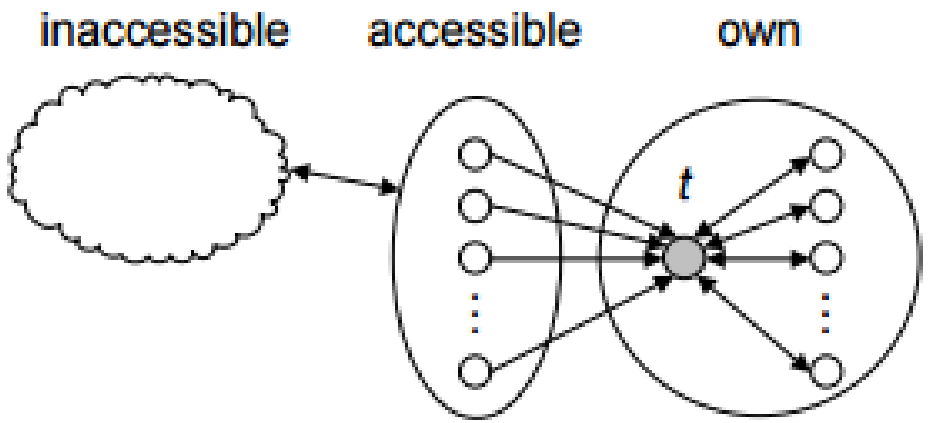
- Repetition
  - Of some specific terms
  - False metadata
  - → fool TF-IDF
- Dumping
  - High number of terms
  - Dictionaries

Web spam with links

- Three types of pages
  - Inaccessible
  - Accessible
    - Comments, blogs
    - Spammer can post links toward target page
  - Own pages
    - Fully controlled by spammer
    - On potentially many domains

Web spam with links

- N number of pages and M number of pages of the spammer.
- Rank contribution by accessible pages x
- Target page pagerank score y
- Rank of each page of the farm is $\frac{\beta y}{M} + \frac{1-\beta}{N}$
- $y = \frac{x}{1-\beta^2} + \frac{\beta}{1+\beta} \cdot \frac{M}{N}$



Key notions

- HITS algorithm
- Pagerank algorithm
- Stationary distribution