Procesado de grafos y Spark

2021 v2

Paulo Villegas

Índice

Parte I: Introducción a grafos

- Principios básicos sobre grafos
- Ejemplos
- Taxonomía de redes
- Software para procesado de Grafos

Parte II: Teoría de grafos

- El valor de una red
- Small worlds & power laws
- Dinámica de redes

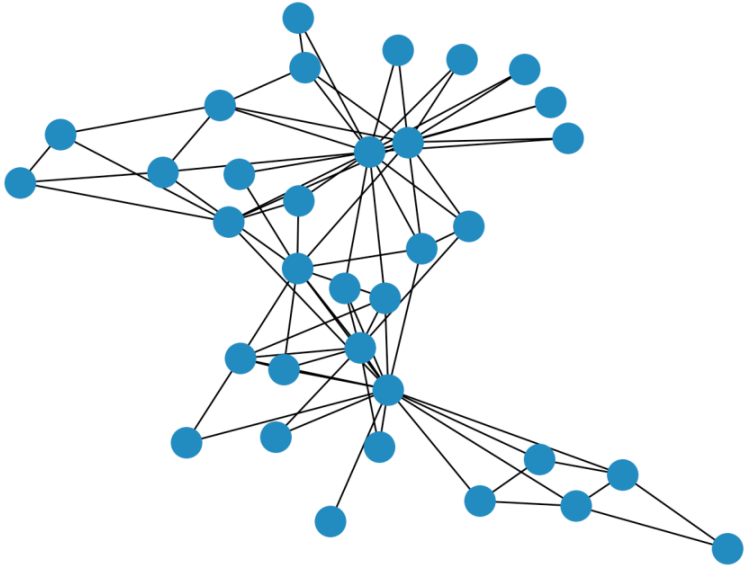
Índice

Parte III: Grafos y Spark

- GraphX
- GraphFrames
  - Instalación
  - Funcionalidad básica
  - Algoritmos

Parte del material procede de:

- Carlos del Cacho, UAM
- *Networks and Social Networks*, Dr. C. Lee Giles, Penn State University



Parte I: introducción a grafos

Paulo Villegas

Principios básicos de grafos

Conceptos básicos

Grafo ⟷ **Red** → **Red social**

matemáticas

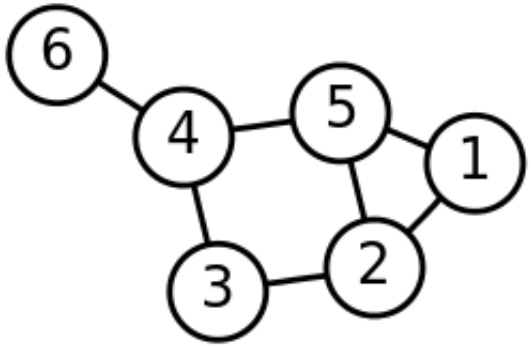
Teoría de redes (comunicaciones, informática)

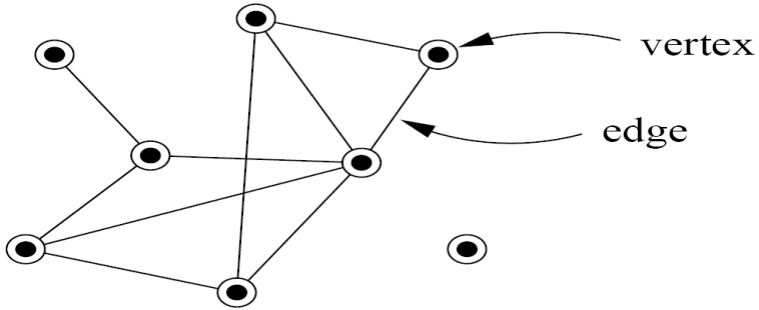
SNA
(social network analysis)

Algunas definiciones básicas

- **Nodo, vértice:** elemento componente fundamental de la red
  - site (physics), node (computer science), or actor (sociology)

- **Enlace, arista:** conexión existente entre dos nodos
  - bond (physics), link or edge (computer science), or tie (sociology)



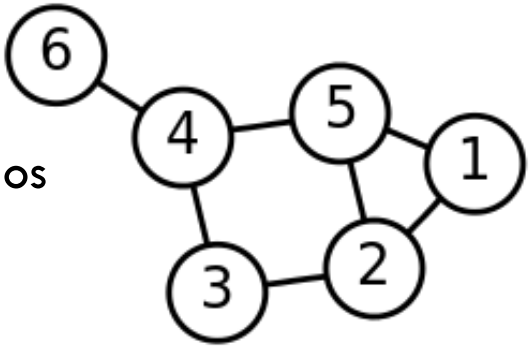


Algunas definiciones básicas

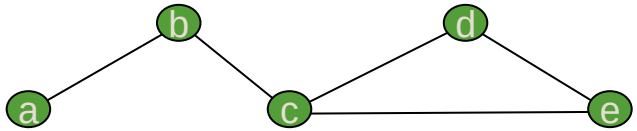
- Un enlace es dirigido *(directed)* si transcurre solo en una dirección, y no dirigido *(undirected)* si transcurre en las dos indistintamente

- Un **grafo dirigido** *(directed graph o digraph)* es aquél que tiene todos sus enlaces dirigidos. Si no, es un **grafo no dirigido**

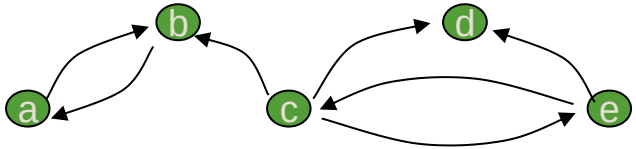
- Un grafo no dirigido puede ser representado por un grafo dirigido en el que los vértices conectados siempre tienen dos enlaces, uno en cada dirección



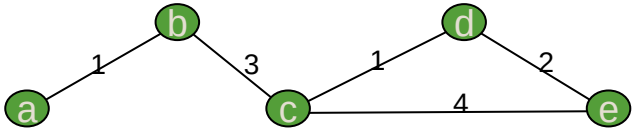
Enlaces binarios/cuantificados, dirigidos/no dirigidos



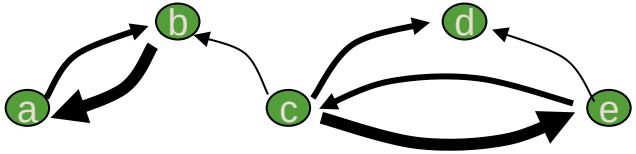
Undirected, binary



Directed, binary



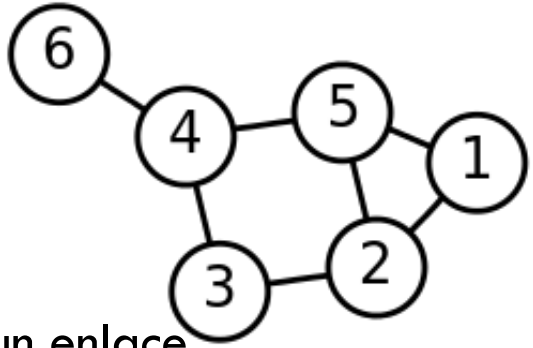
Undirected, Valued



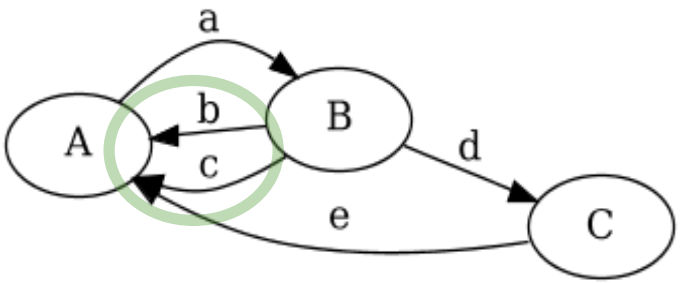
Directed, Valued

Algunas definiciones básicas

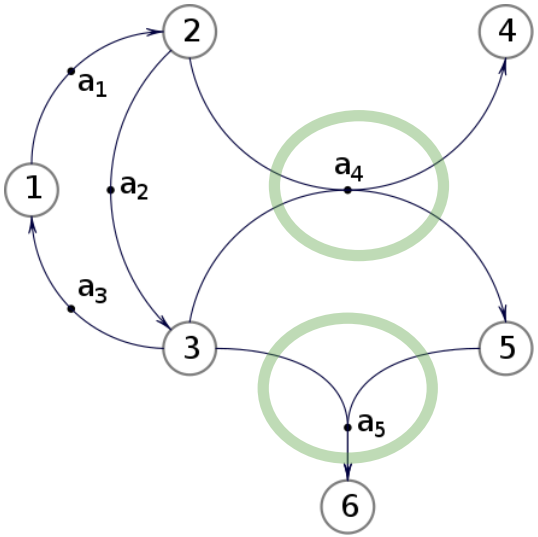
- **Grado** de un nodo: número de enlaces que conectan al nodo
  - No es necesariamente el número de vértices conectados al nodo, ya que puede haber más de un enlace entre dos nodos (multigrafo).
  - Un grafo dirigido tiene para cada enlace un grado de entrada (*in-degree*) y uno de salida (*out-degree*): número de enlaces entrantes y salientes.
    - **Grado de entrada** (in-degree)
    - **Grado de salida** (out-degree)



- Casos especiales
  - **Multigrafo:** entre dos nodos puede haber más de un enlace
  - **Hipergrafo:** un enlace puede conectar más de dos nodos



multigrafo dirigido



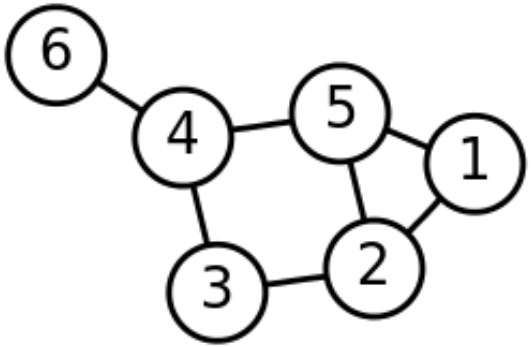
hipergrafo dirigido

a, b, c son enlaces entre A y B
(b y c son ambos enlaces entre A y B, y ambos en la misma dirección)

a_1, … a_n son los enlaces del hipergrafo;
a_4 y a_5 unen más de dos vértices (de forma dirigida)

Algunas definiciones no tan básicas

- **Paseo:** Secuencia de vértices conectados por aristas cada uno con el siguiente (con repeticiones)

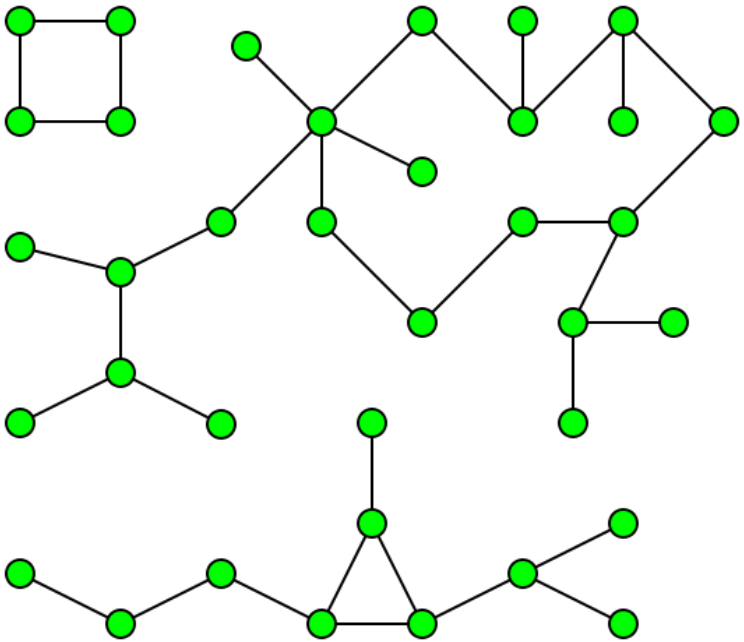


- **Camino (path):** Paseo sin aristas repetidas
- **Orden:** Número de nodos del camino
- **Ciclo (cycle):** Camino que empieza y acaba en el mismo nodo, con al menos tres enlaces
- **Subgrafo:** Un subconjunto de aristas y nodos dentro de un grafo

Algunas definiciones no tan básicas

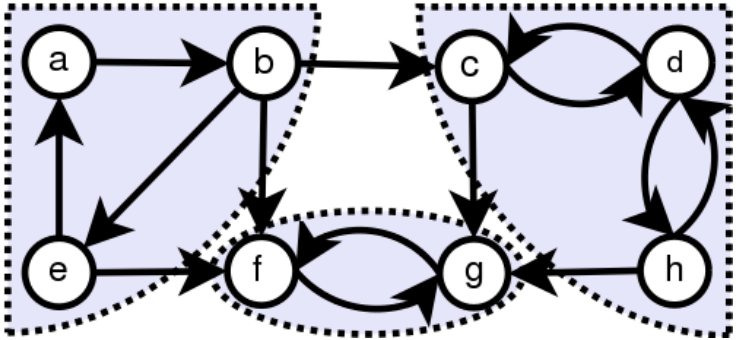
- **Componente conexa** (*connected component*): En un grafo no dirigido, la *componente conectada* de un nodo es el subgrafo compuesto por los vértices que pueden ser alcanzados por caminos.

- En un grafo dirigido, un vértice tiene tanto un componente de entrada *(in-component)* como uno de salida *(out-component)*: los conjuntos de vértices desde los que se puede llegar a él, y aquellos a los que se puede llegar desde él.



Algunas definiciones no tan básicas

- **Componente fuertemente conectado** (*strongly connected component*): para grafos dirigidos, es un conjunto de nodos tales que todos pueden alcanzarse desde cualquiera de los demás.

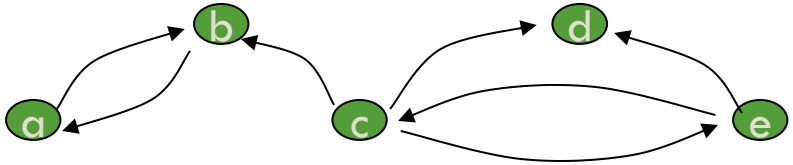


- Si el componente fuertemente conectado es todo el grafo, entonces se dice que el grafo es fuertemente conectado

Algunas definiciones no tan básicas

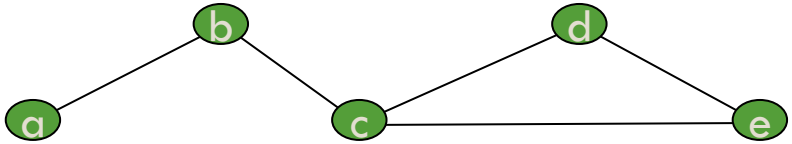
- **Camino geodésico (*geodesic path*):** el camino más corto a través de la red entre dos nodos. Puede no ser único.
- **Diámetro:** la longitud (número de enlaces) del camino geodésico más largo entre cualquier par de vértices.

de grafos a matrices



Directed, binary

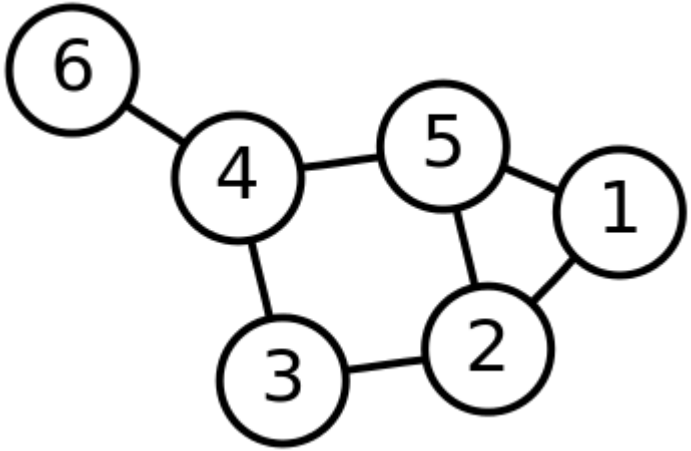
	a	b	c	d	e
a		1			
b	1				
c		1		1	1
d					
e			1	1	



Undirected, binary

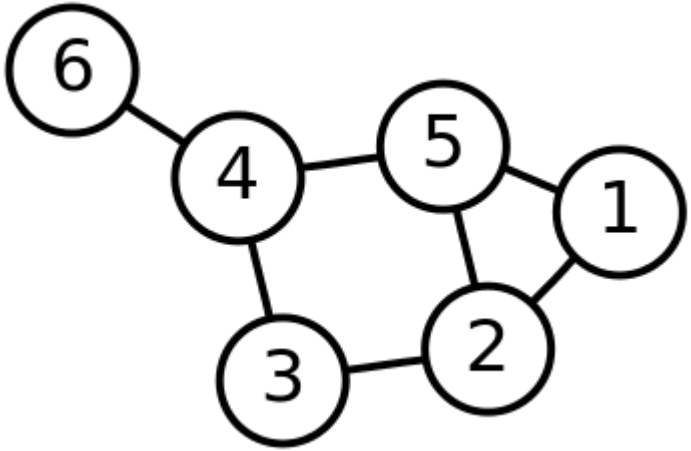
	a	b	c	d	e
a		1			
b	1		1		
c		1		1	1
d			1		1
e			1	1	

Matriz de adyacencia



$$
\begin{array}{c|cccccc}
 & 1 & 2 & 3 & 4 & 5 & 6 \\
\hline
1 & 0 & 1 & 0 & 0 & 1 & 0 \\
2 & 1 & 0 & 1 & 0 & 1 & 0 \\
3 & 0 & 1 & 0 & 1 & 0 & 0 \\
4 & 0 & 0 & 1 & 0 & 1 & 1 \\
5 & 1 & 1 & 0 & 1 & 0 & 0 \\
6 & 0 & 0 & 0 & 1 & 0 & 0
\end{array}
$$

Matriz de grado



$$
\begin{array}{c|cccccc}
 & 1 & 2 & 3 & 4 & 5 & 6 \\
\hline
1 & 2 & 0 & 0 & 0 & 0 & 0 \\
2 & 0 & 3 & 0 & 0 & 0 & 0 \\
3 & 0 & 0 & 2 & 0 & 0 & 0 \\
4 & 0 & 0 & 0 & 3 & 0 & 0 \\
5 & 0 & 0 & 0 & 0 & 3 & 0 \\
6 & 0 & 0 & 0 & 0 & 0 & 1
\end{array}
$$

de matrices a listas

	a	b	c	d	e
a		1			
b	1		1		
c		1		1	1
d			1		1
e			1	1	

Adjacency List

a	b
b	a c
c	b d e
d	c e
e	c d

Arc List

a b
b a
b c
c b
c d
c e
d c
d e
e c
e d

Representación de matrices dispersas

- De la matriz de grado de la pag 17:

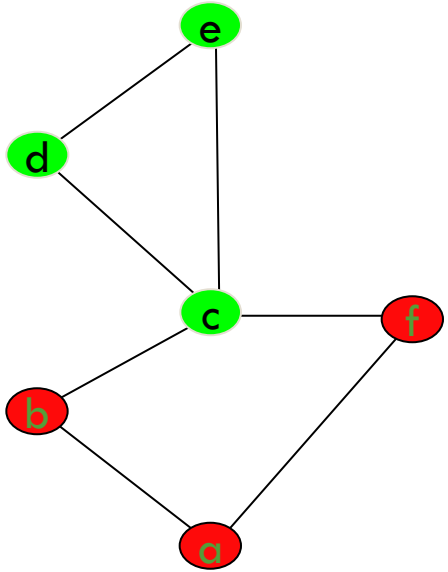
Índice de Fila	Índice de Columna	Valor
1	1	2
2	2	3
3	3	2
…		

- Cuando no aparece un valor para una relación [índice,columna] se supone que es cero

Cálculos con matrices

Las matrices permiten calcular patrones de mezclado: conexiones entre nodos de distintos tipos. Para ello se multiplica la matriz de adyacencia por un indicador de categoría.

N° enlaces de cada nodo con nodos de cada clase



	a	b	c	d	e	f
a	0	1	0	0	0	1
b	1	0	1	0	0	0
c	0	1	0	1	1	1
d	0	0	1	0	1	0
e	0	0	1	1	0	0
f	1	0	1	0	0	0

$\times$

Race	
1	0
1	0
0	1
0	1
0	1
1	0

$=$

XRace	
2	0
1	1
2	2
0	2
0	2
1	1

$$\mathbf{Race}^t \cdot \mathbf{XRace} =$$

	R	G
R	4	2
G	2	6

Cálculos con matrices

Las operaciones con matrices permiten comprobar la dirección de los enlaces, y distinguir enlaces simétricos y asimétricos

Un grafo asimétrico puede convertirse en simétrico sumando su matriz transpuesta

$$
\begin{array}{c}
\mathbf{x} \\
\begin{matrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \end{matrix}
\end{array}
+
\begin{array}{c}
\mathbf{x}^{\mathbf{T}} \\
\begin{matrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \end{matrix}
\end{array}
=
\begin{matrix} 0 & 2 & 0 & 0 & 0 \\ 2 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 2 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 2 & 1 & 0 \end{matrix}
$$

Grado de un nodo a partir de matrices

- outdegree = $\sum_{i=1}^{n} A_{ij}$

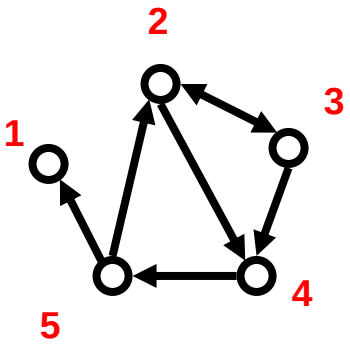
grado de salida para el nodo 3 => 2 $\sum_{j=1}^{n} A_{3j}$

(suma de valores de la 3ª fila)

- indegree = $\sum_{j=1}^{n} A_{ij}$

grado de entrada para el nodo 3 => 1 $\sum_{i=1}^{n} A_{i3}$

(suma de valores de la 3ª columna)

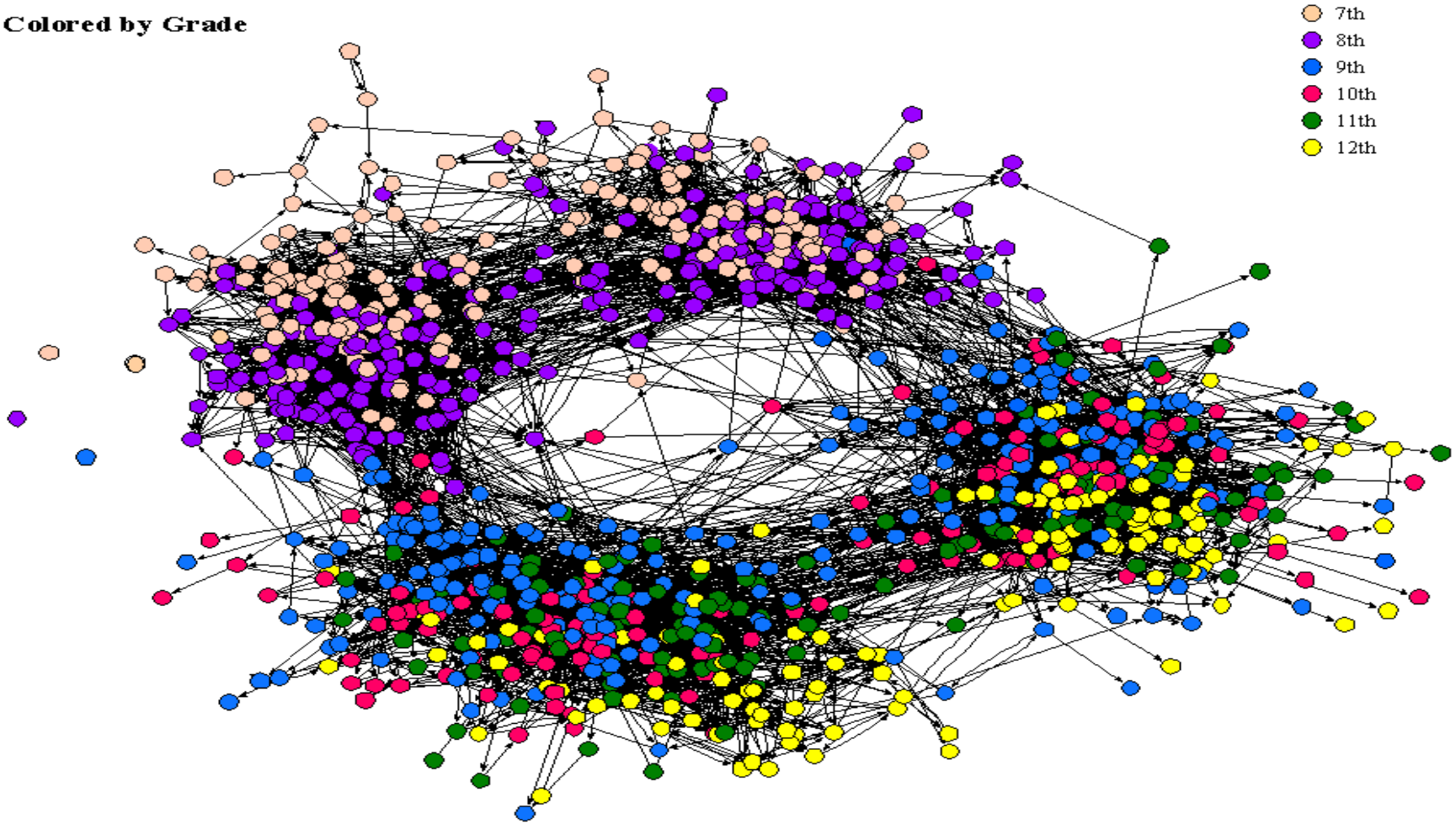


$$A = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 \end{pmatrix}$$

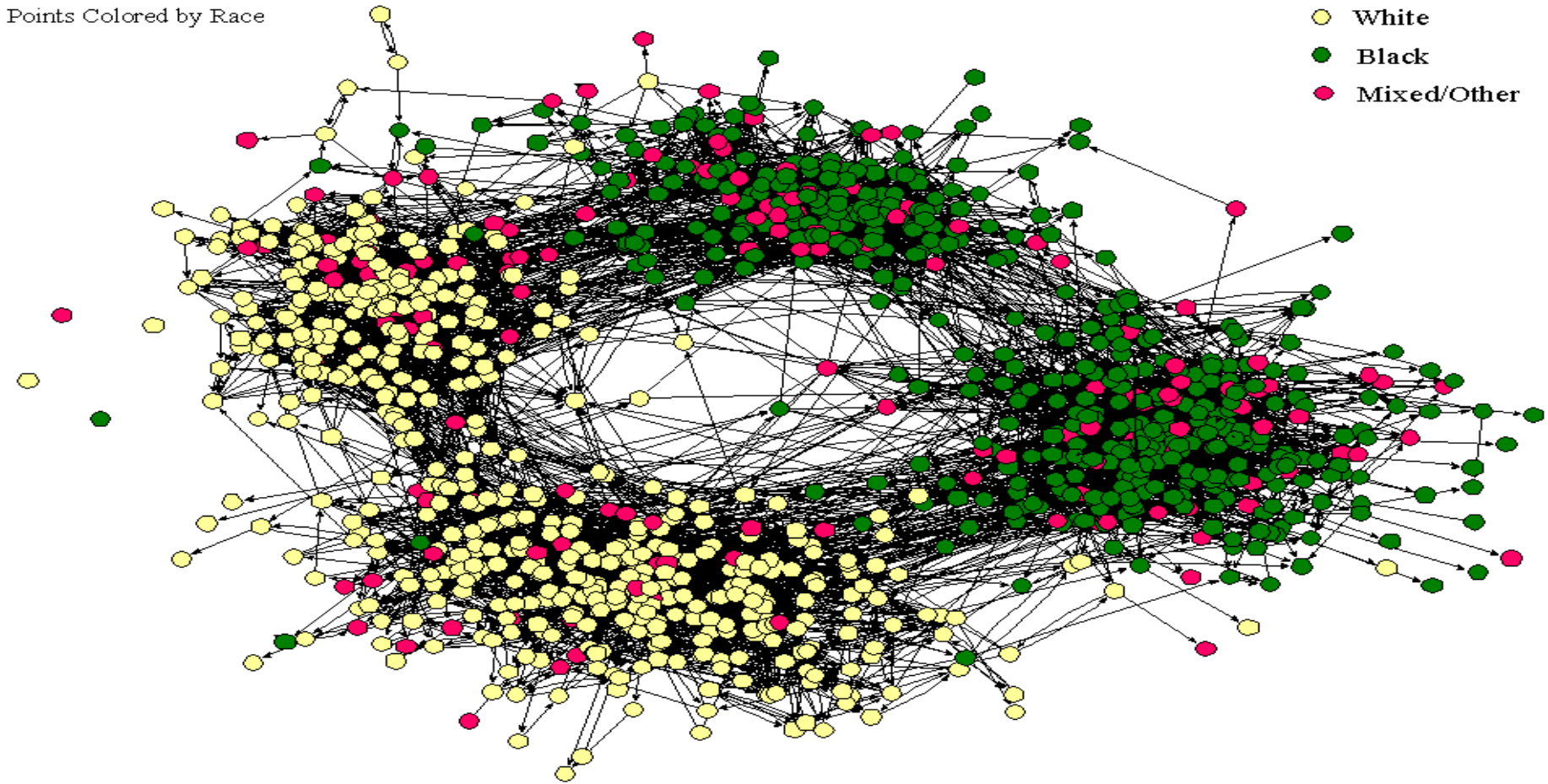
$$A = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 \end{pmatrix}$$

Ejemplos de redes

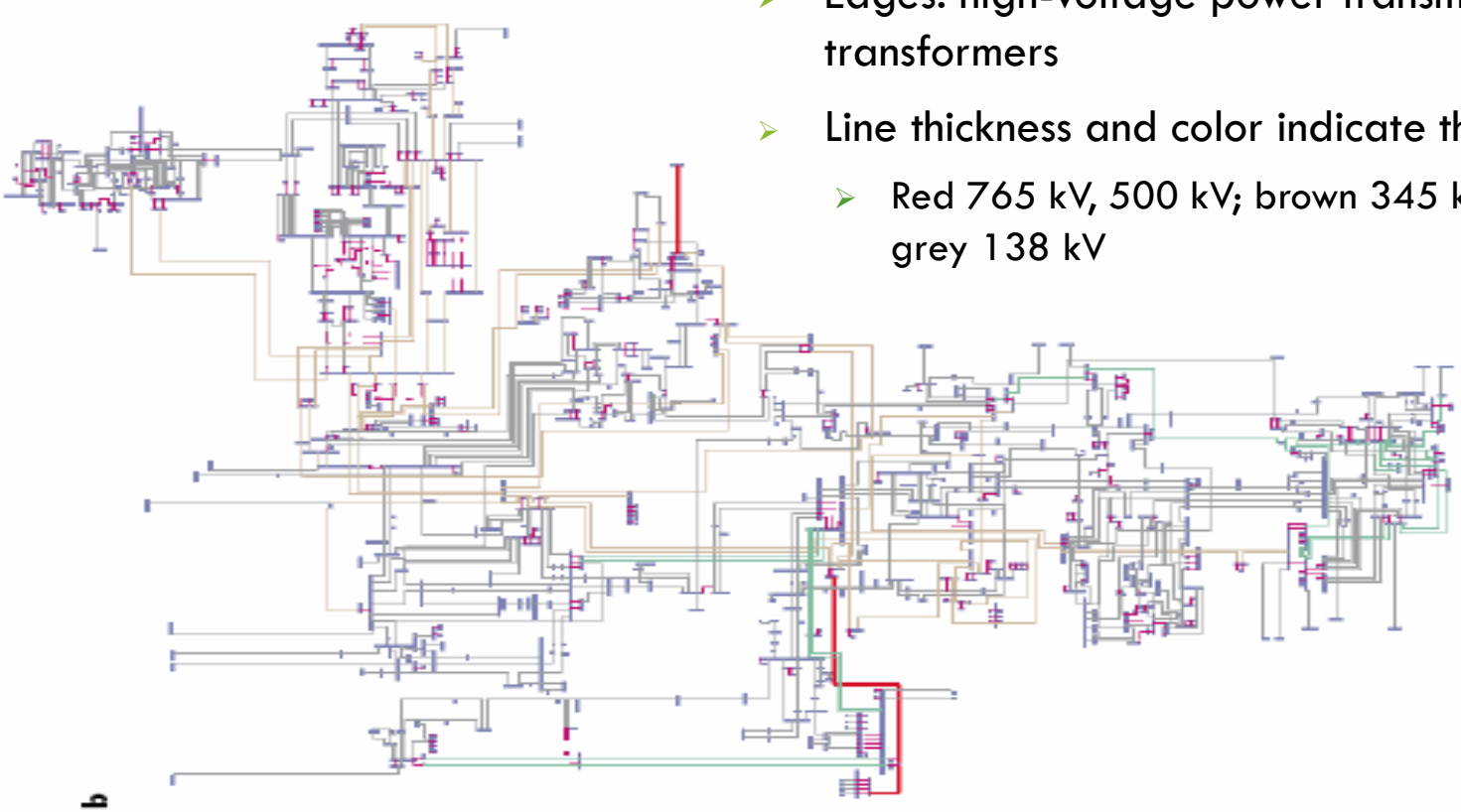
The Social Structure of "Countryside" School District



The Social Structure of "Countryside" School District



New York State Power Grid



- Vertices: generators and substations
- Edges: high-voltage power transmission lines and transformers
- Line thickness and color indicate the voltage level
  - Red 765 kV, 500 kV; brown 345 kV; green 230 kV; grey 138 kV

Redes de comunicaciones

nodos y enlaces

Líneas de datos

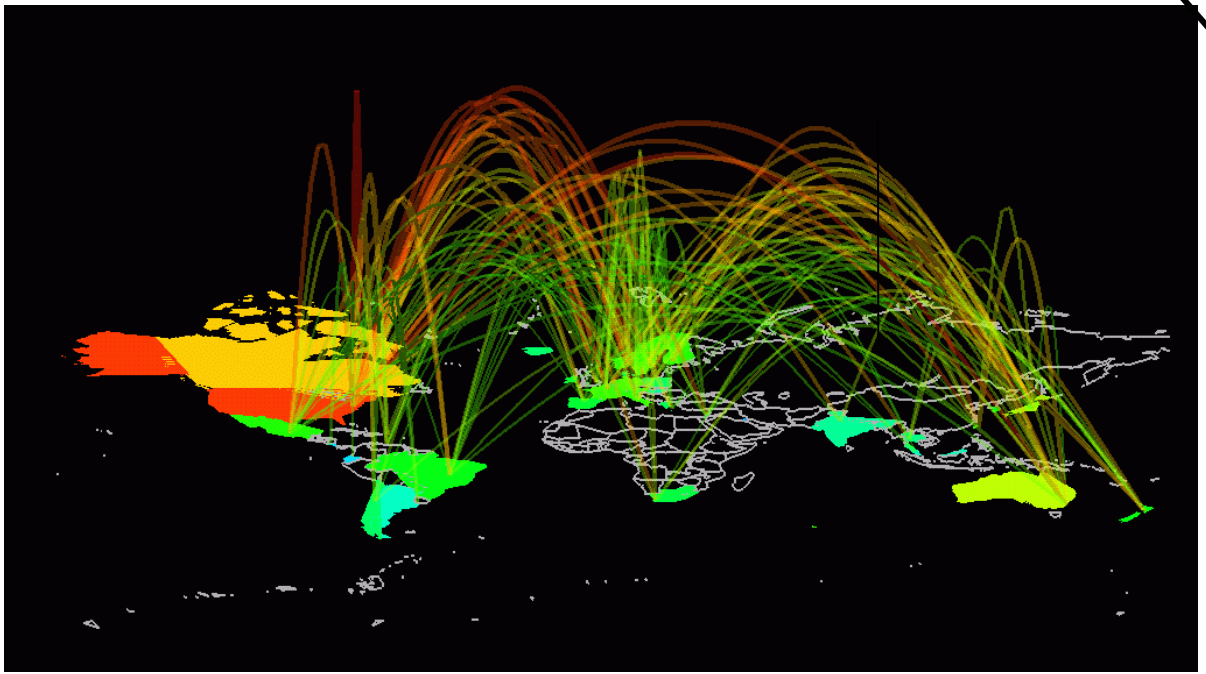
redes de cable

redes de difusión

-ordenadores

-routers

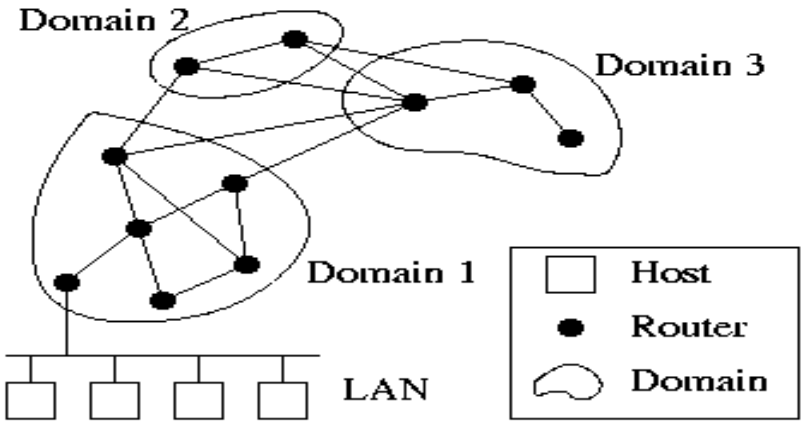
-satélites

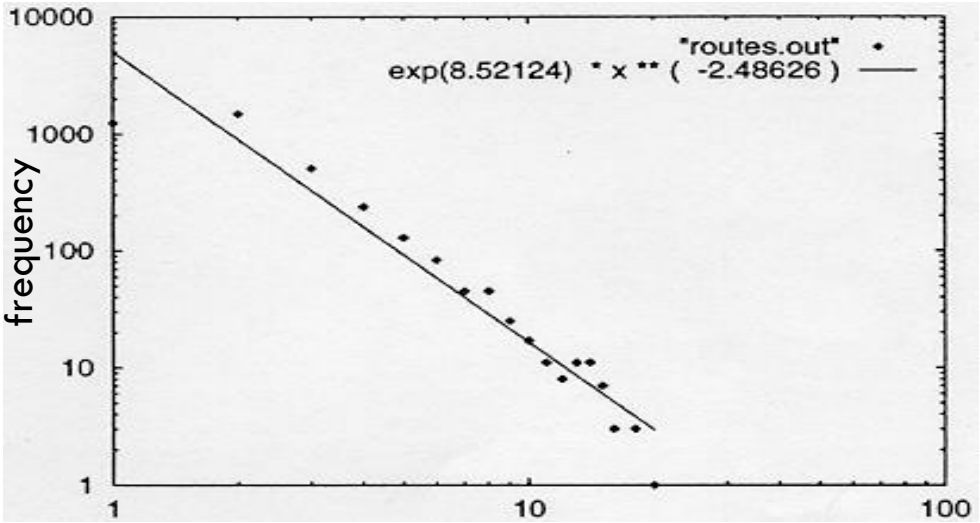


Communication networks: Many non-identical components with diverse connections between them.

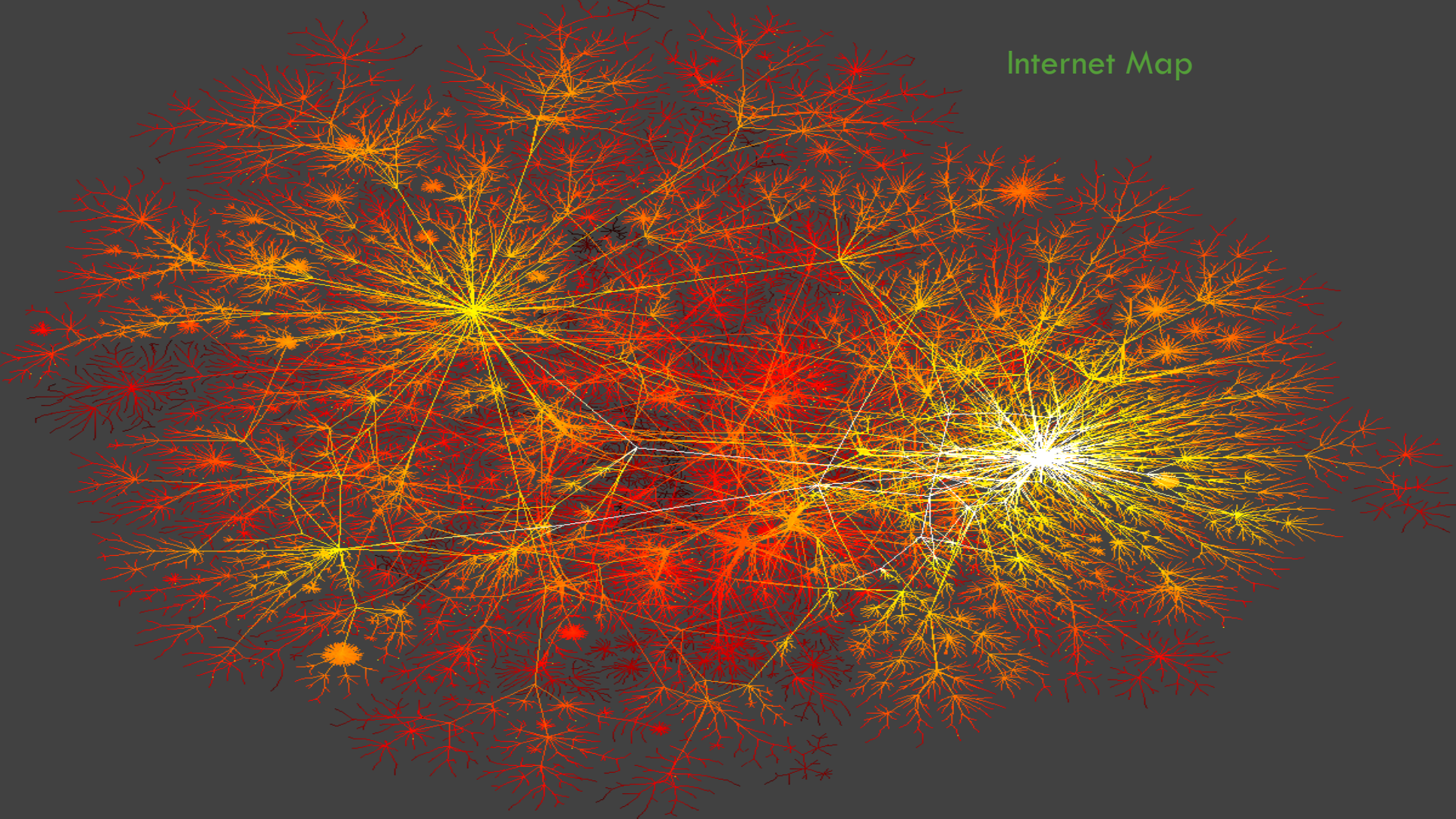
Internet Backbone

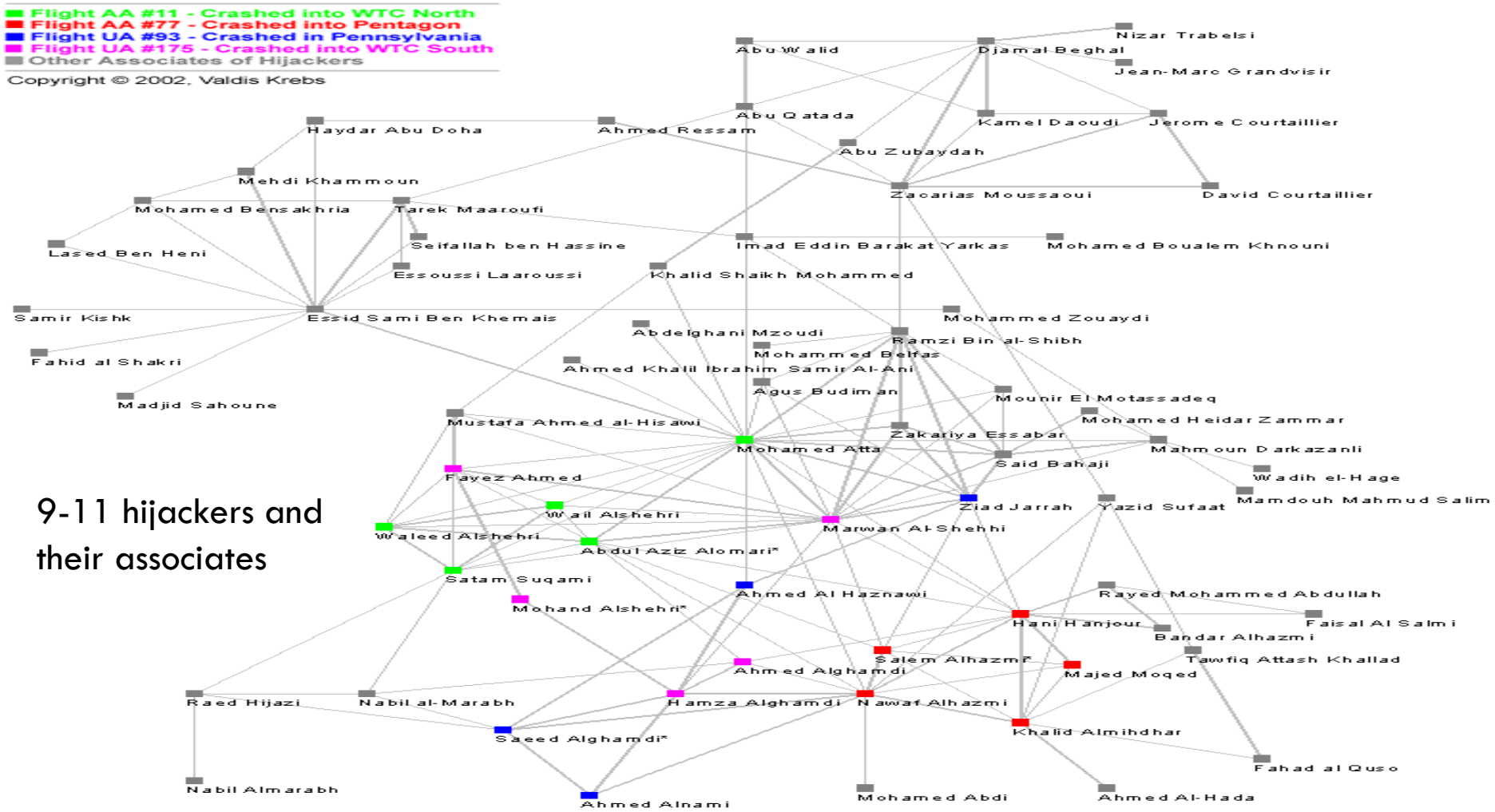
Nodes: computers, routers
Links: physical lines





(Faloutsos, Faloutsos and Faloutsos, 1999)

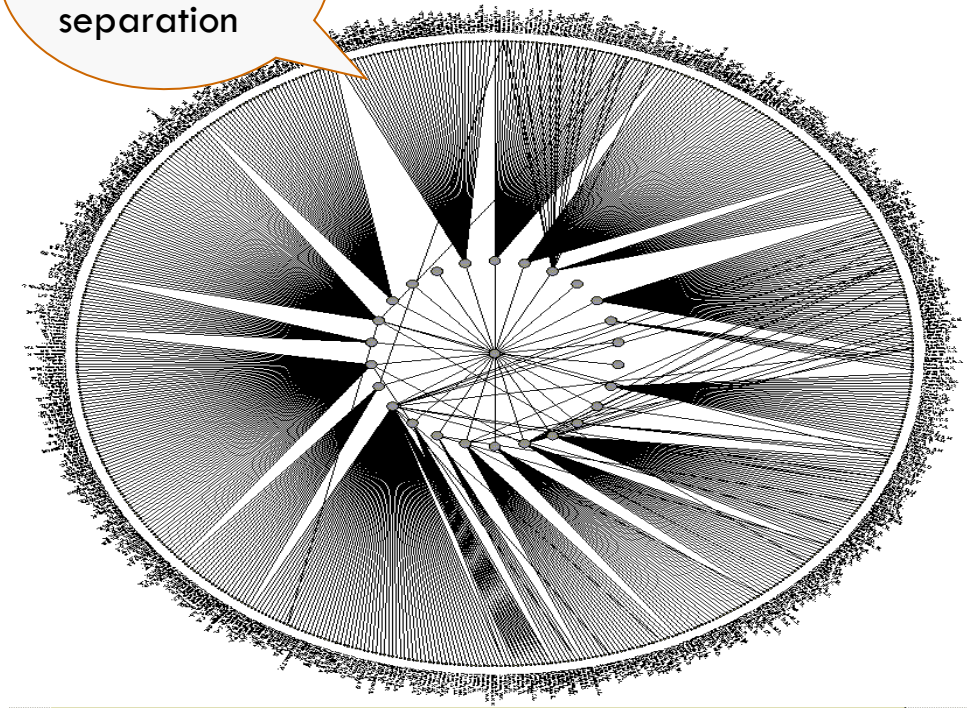

Internet Map



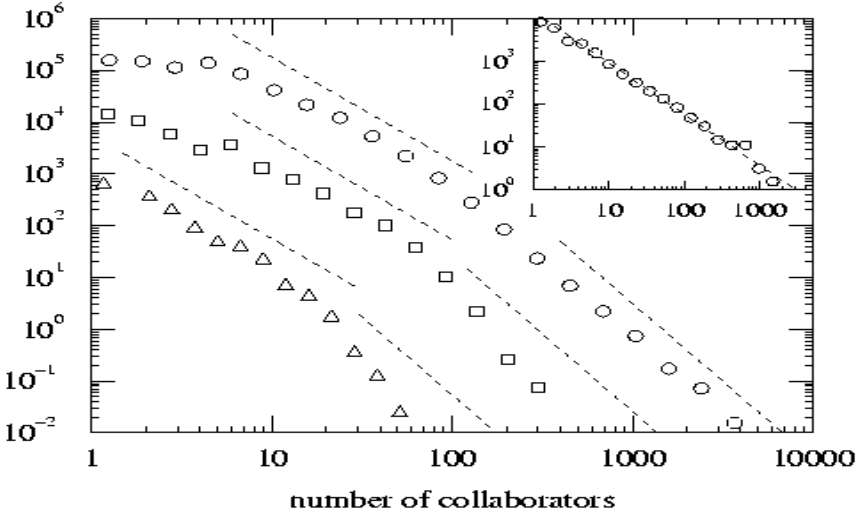
9-11 hijackers and their associates

Science Co-authorship

map of 3 degrees of separation



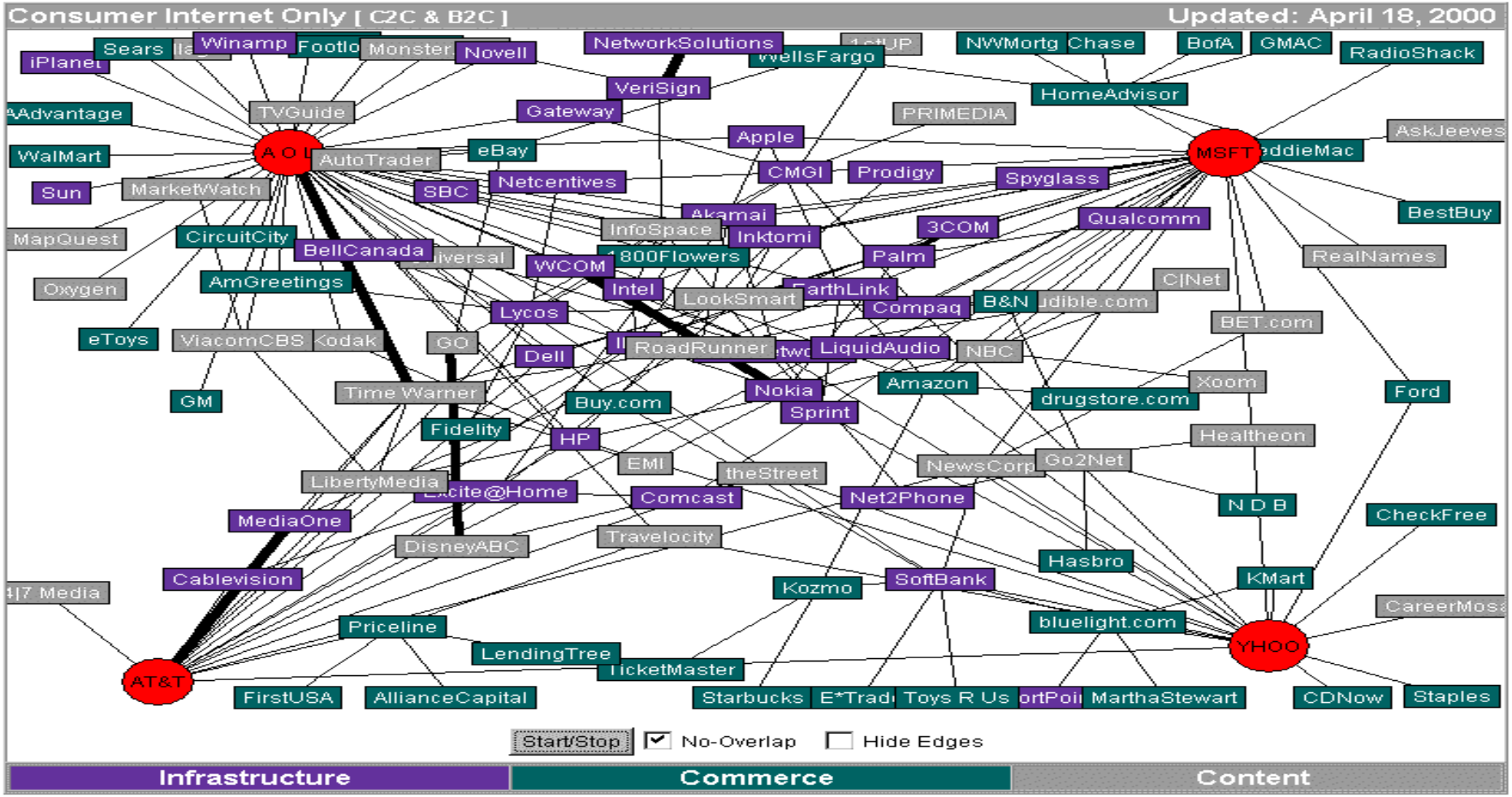
Nodes: scientist (authors)
Links: write paper together



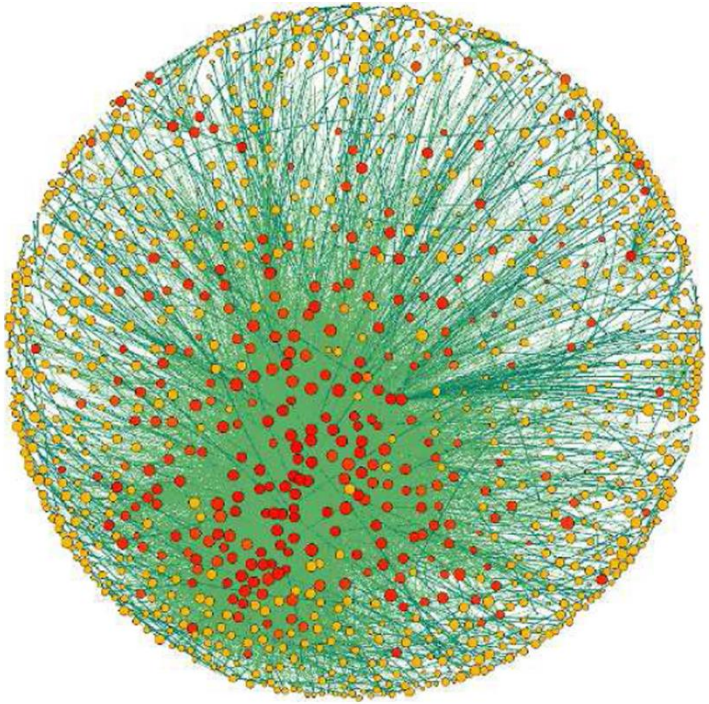
(Newman, 2000, H. Jeong et al 2001)

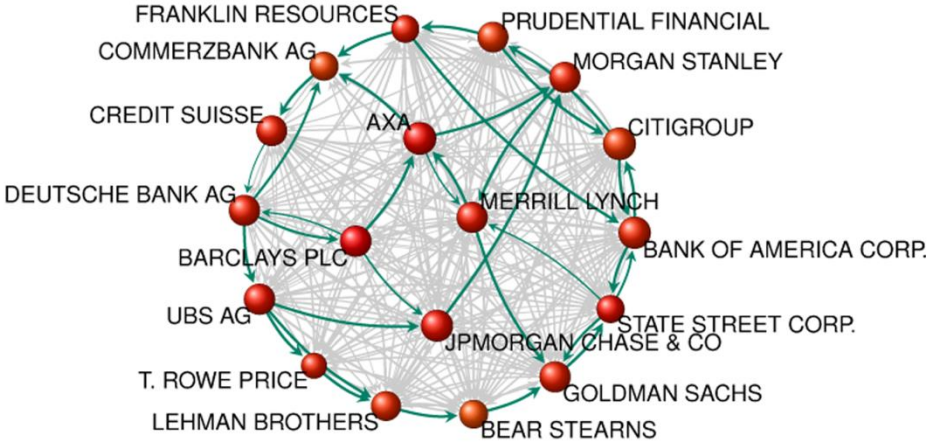
Internet Industry *Strategic Alliances, Joint Ventures, and Other Partnerships*

Printable Map and Network Metrics also available. Copyright © 2000, Valdis Krebs



S. Vitali, J. B. Glattfelder, S. Battiston, *The Network of Global Corporate Control*, 2011

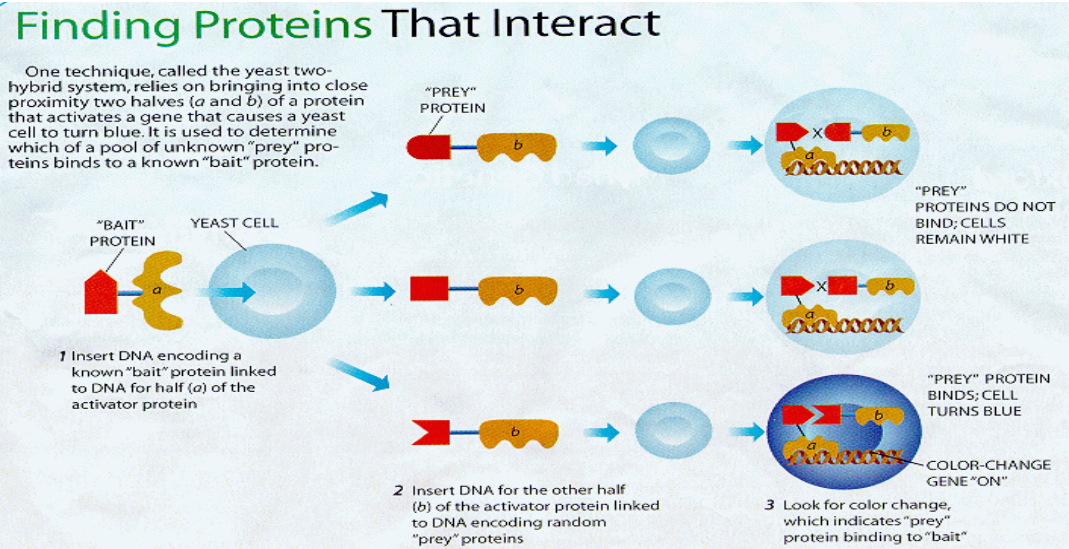




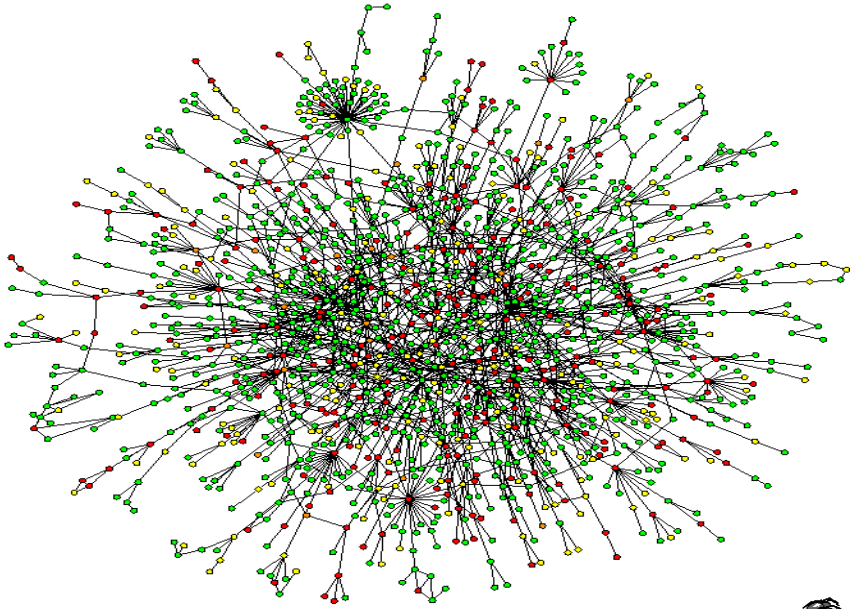
Protein Interaction map

Nodes: proteins

Links: physical interactions (binding)



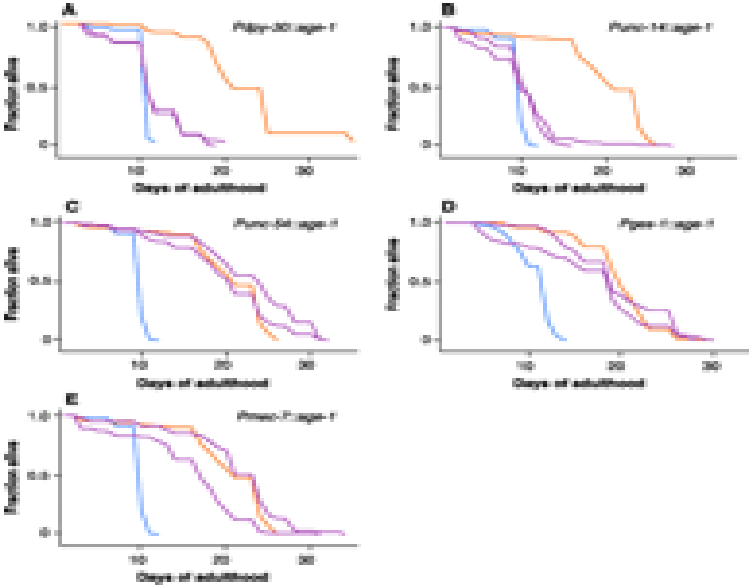
Yeast Protein Network

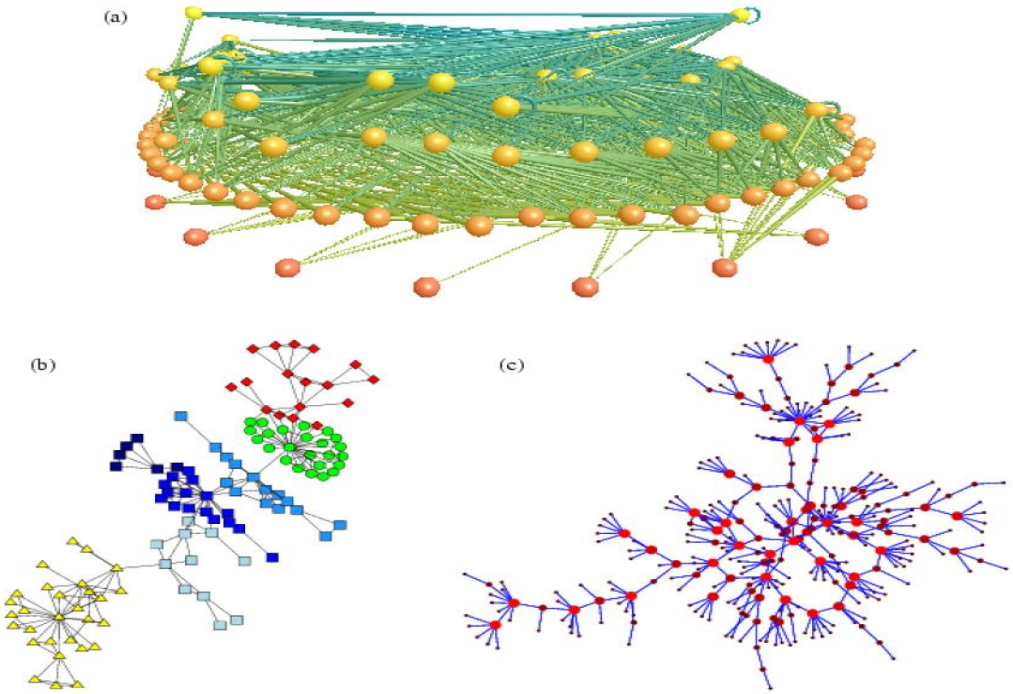


P. Uetz, *et al.* *Nature* **403**, 623-7 (2000).

Caenorhabditis Elegans nervous system

- Vertices: neurons in the *C. elegans* worm
- Edges: axons/synapses between neurons





a) A food web of predator-prey interactions between species in a freshwater lake

b) The network of collaborations between scientists at a private research institution

c) A network of sexual contacts between individuals

[Newman, 2003]

Taxonomía de redes

Taxonomía de redes

Es posible agrupar una gran cantidad de las redes en 4 grupos: [Newman 2003]

1) **Social networks:** set of people or groups of people with some pattern of contacts or interactions between them
   - the patterns of friendships between individuals
   - business relationships between companies
   - intermarriages between families

2) **Information networks**
   - network of citations between academic
   - the World Wide Web
   - preference networks

Taxonomía de redes

3) **Technological networks**
   - electric power grid
   - network of airline routes
   - Internet

4) **Biological networks**
   - network of metabolic pathways
   - the genetic regulatory network
   - the food web
   - neural networks

Clasificación por niveles de estudio

We can examine networks across multiple levels:

1. *Ego-network*
   - Have data on a respondent (*ego*) and the people they are connected to (*alters*).
   - May include estimates of connections among alters (*ego-network with alters*)

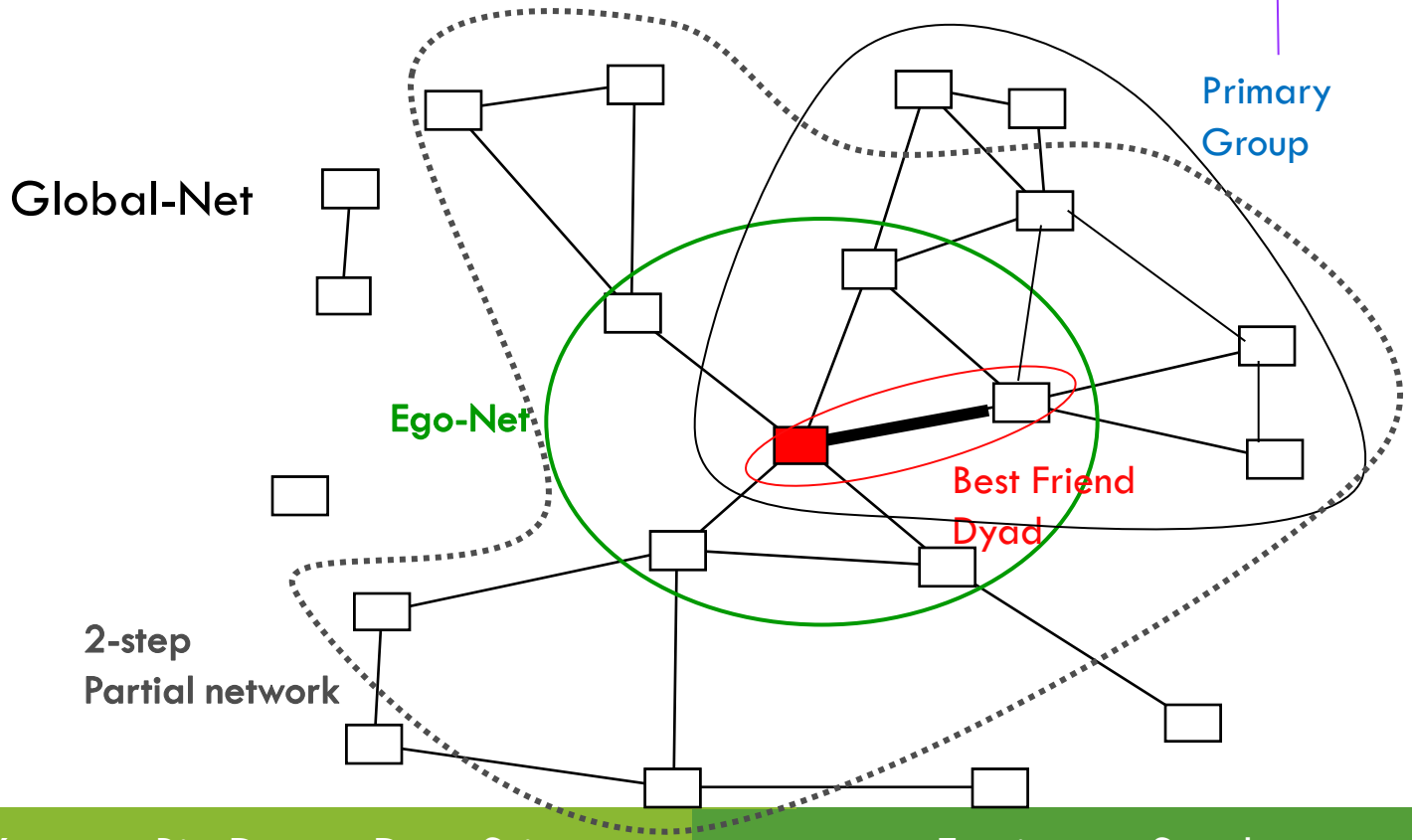
2. *Partial network*
   - Ego networks plus some amount of tracing to reach contacts of contacts
   - Something less than full account of connections among all pairs of actors in the relevant population

Clasificación por niveles de estudio

3. *Full networks: Complete or "Global" data*
   - Data on all actors within a particular (relevant) boundary.
   - Never exactly complete (due to missing data), but boundaries are set
   - Example: Co-authorship data among all writers in the social sciences, friendships among all students in a classroom

Niveles de estudio

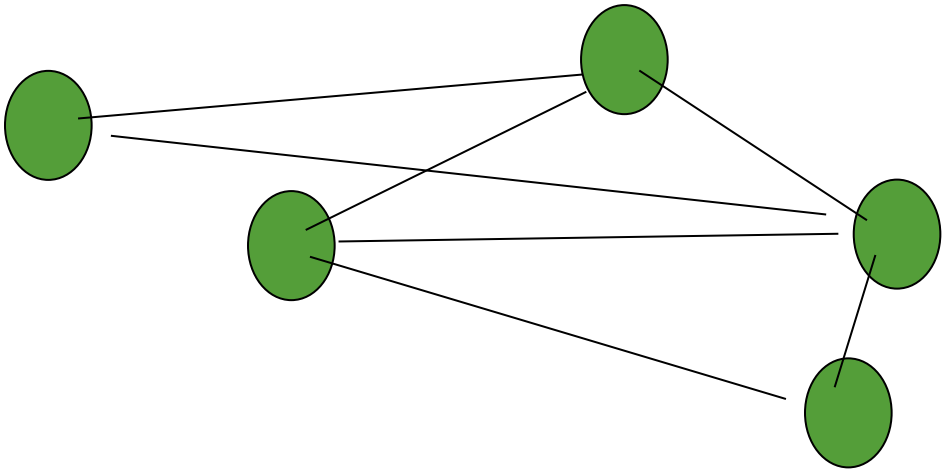
"grupo primario" es un concepto social (relacionado con cohesión de grupo) más que un concepto de red



Representación gráfica de redes

Representación gráfica

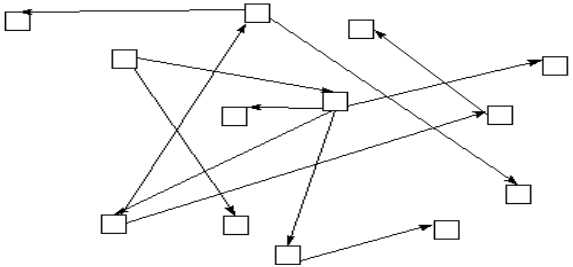
La representación estándar de un grafo (sociograma en SNA) mapea la estructura del grafo en espacio 2D/3D, uniendo los nodos con líneas

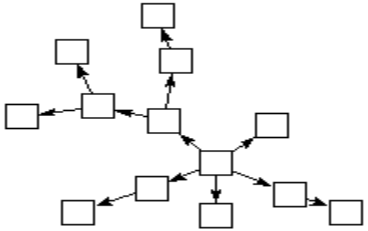


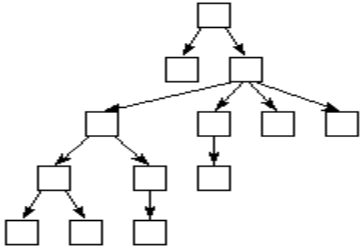
Representación gráfica

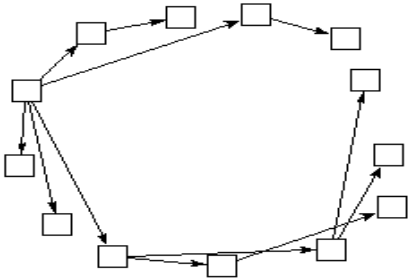
No existe una única forma de visualizar un grafo

Todas estas gráficas corresponden a la misma red





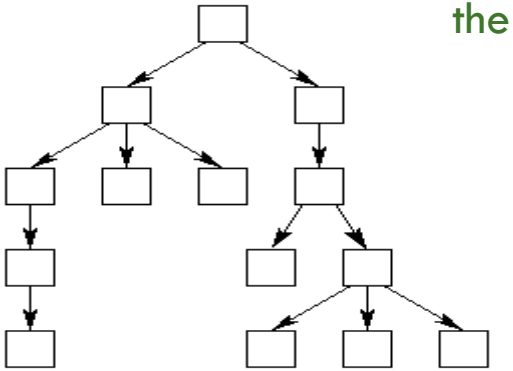




Representación gráfica

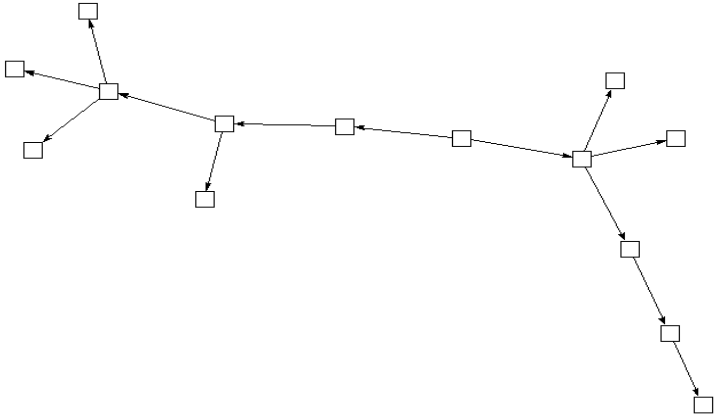
Una visualización ayuda a obtener una idea sobre el grafo, pero hay que tener siempre presente la influencia del algoritmo de representación usado

Tree-Based layouts



Two images of the same network

Spring-embedder layouts

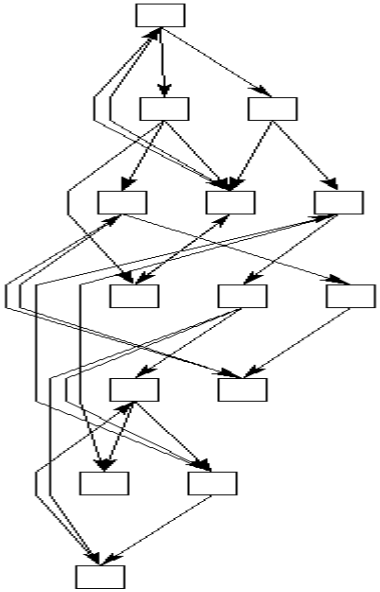


Most effective for very sparse, regular graphs. Very useful when relations are strongly directed, such as organization charts, internet connections, etc

Most effective with graphs that have a strong community structure (clustering, etc). Provides a very clear correspondence between social distance and plotted distance

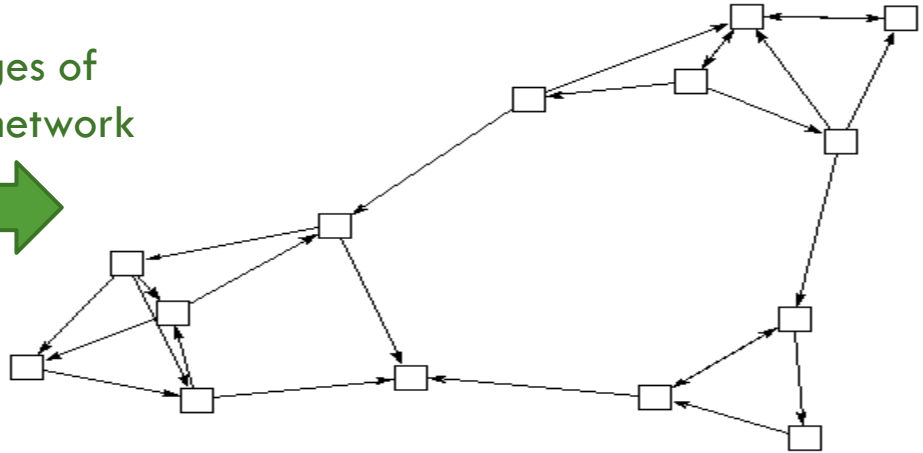
Representación gráfica

Tree-Based layouts



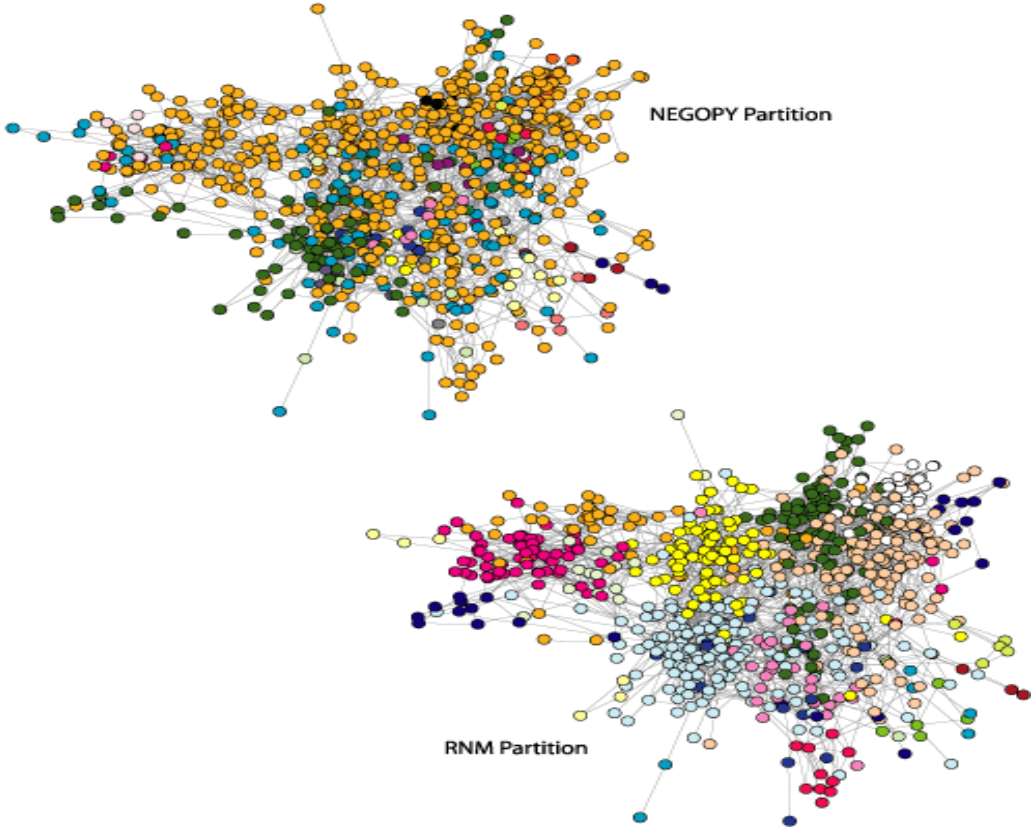
Two images of
the same network

Spring-embedder layouts



El uso de colores o formas para marcar atributos de los nodos facilita la visualización de la distribución de esos atributos.

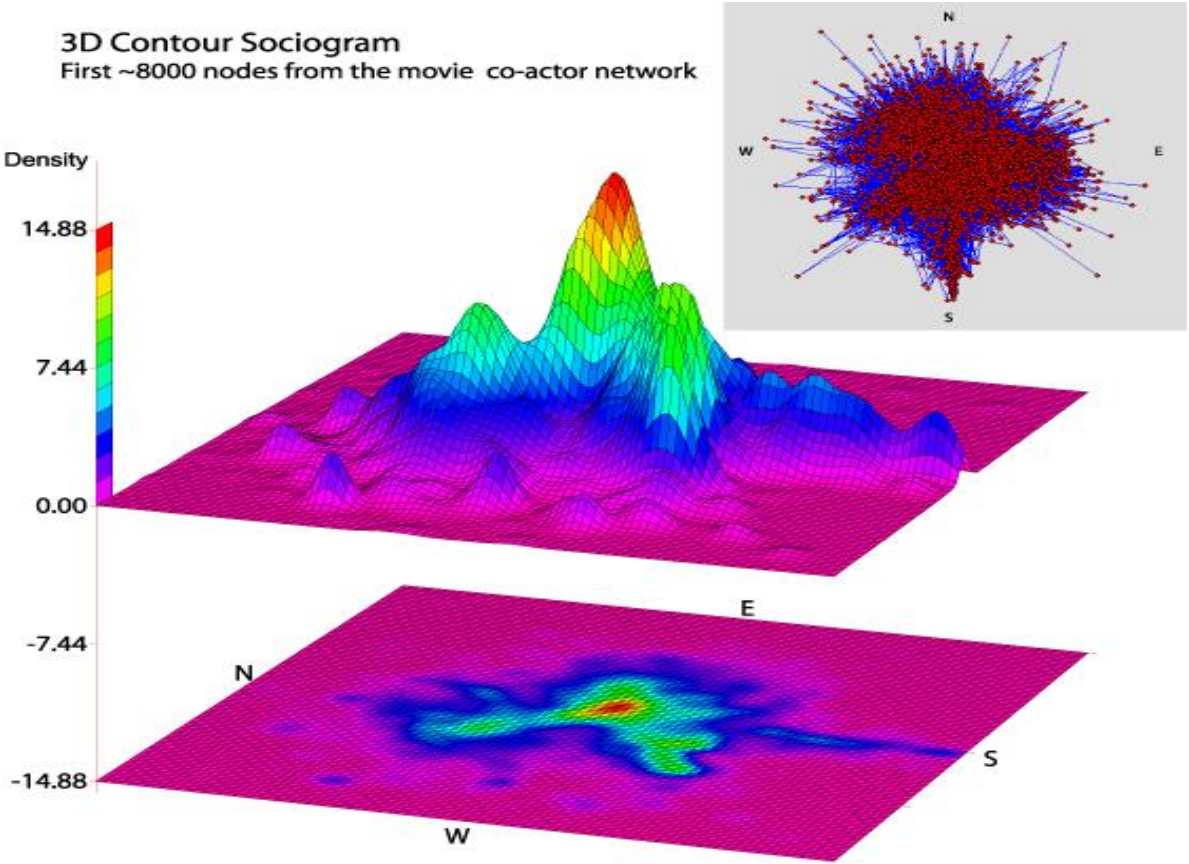
Ejemplo: dos tipos de agrupaciones en una red social escolar



Cuando las redes aumentan de tamaño, la efectividad de los diagramas de grafos decrece, al quedar demasiado complicados para las dimensiones disponibles

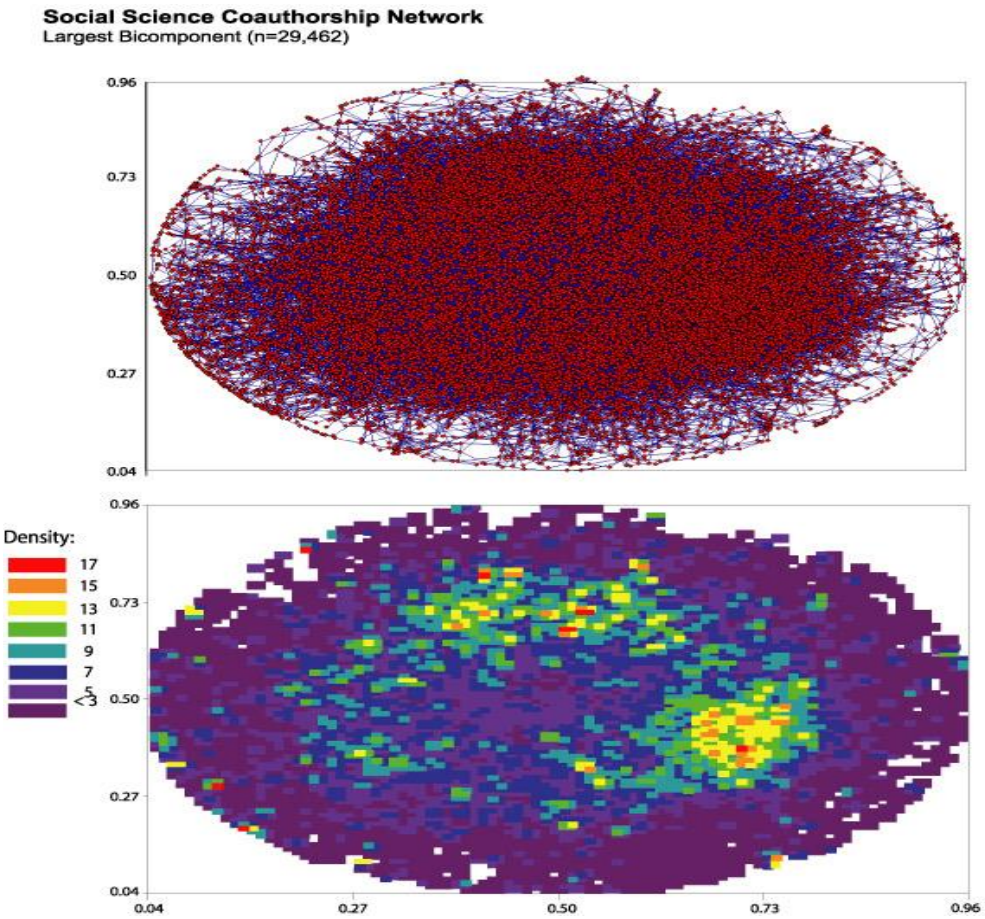
En esos casos puede ser útil utilizar agregados, funciones de densidad, etc, para capturar la esencia de la estructura

Ejemplo: clustering en una red de coparticipación en películas



Ejemplo: grafo de coautoría para ~ 30000 autores

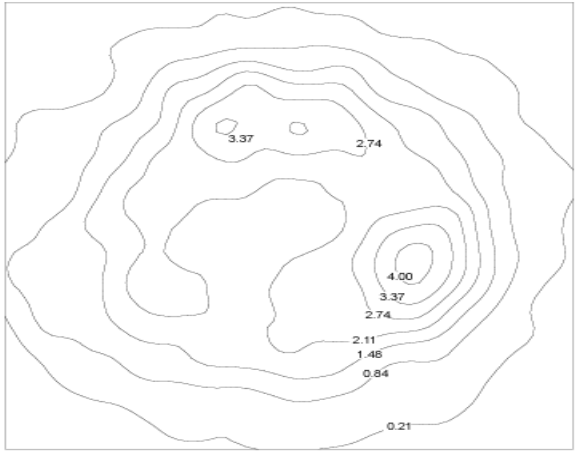
Las dos regiones de alta densidad corresponden a dos temas distintos.



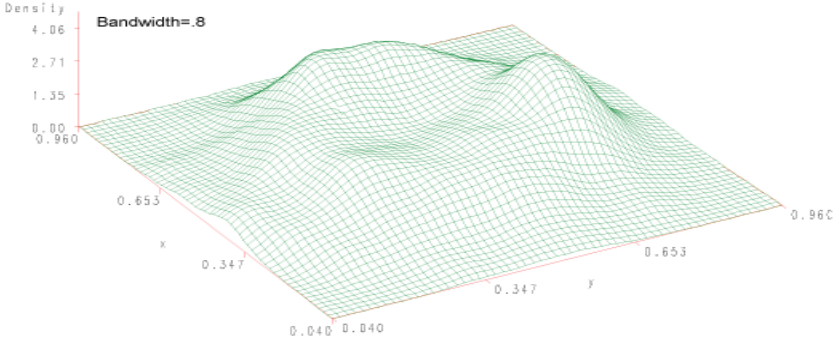
Ejemplo: grafo de coautoría para ~ 30000 autores

Las dos regiones de alta densidad corresponden a dos temas distintos.

Figure 5. Social Science Coauthorship Network
Largest Bicomponent (n=29,462)



Contour Sociogram. Contour values denote bivariate density estimates, indicating every 15th percent of the range between the 5th and 95th percentiles.



Software de procesado de redes

Procesado de grafos

- **Monopuesto**
  - NetworkX (Python)
  - igraph (Python/R/C++)
  - graph-tool (Python)
- **Procesado masivo**
  - Spark (GraphX, GraphFrames)
  - Apache Giraph
  - GraphLab Create/Turi
  - Gelly (Apache Flink)

Graph DB

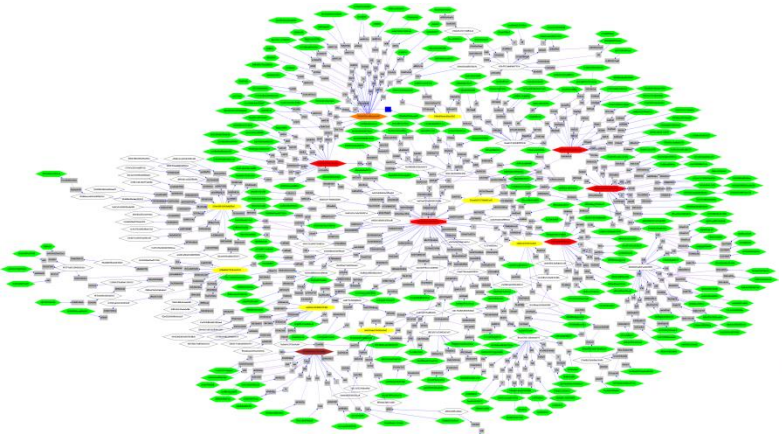
- Neo4j
- OrientDB
- **Semantic Web**
  - AllegroGraph
  - Ontotext GraphDB
  - OpenLink Virtuoso
- **Extensiones a RDBMS estándar: Oracle, SQLServer**

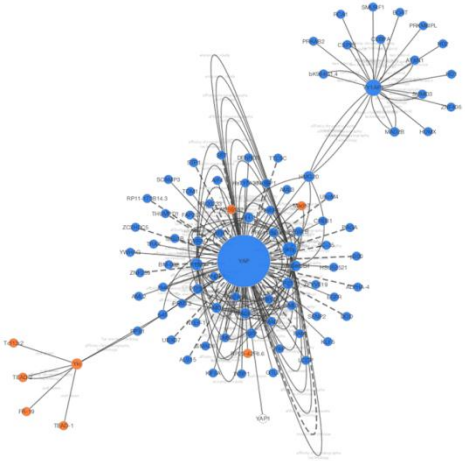
Visualización

Graphviz

Gephi

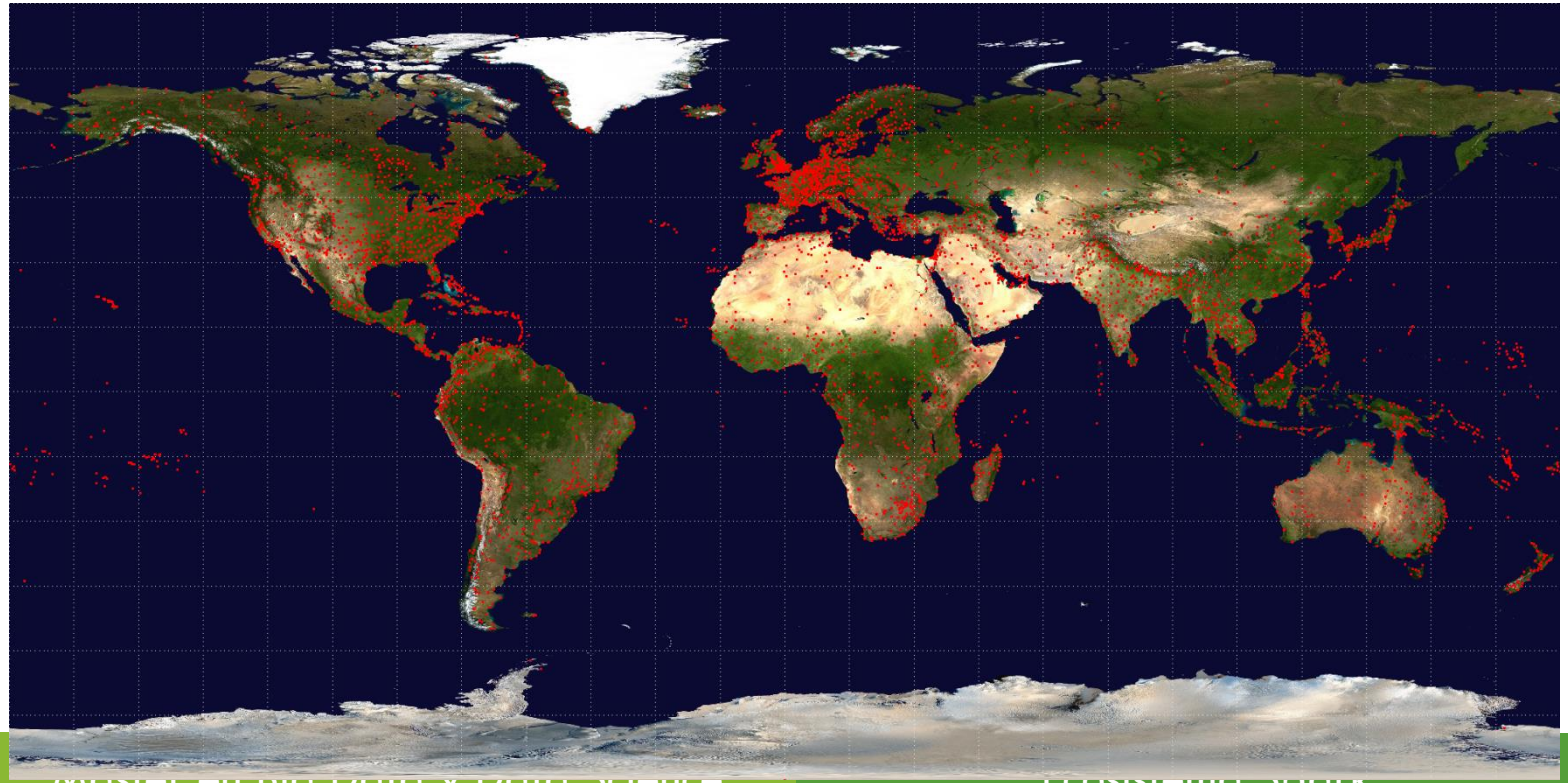
Cytoscape

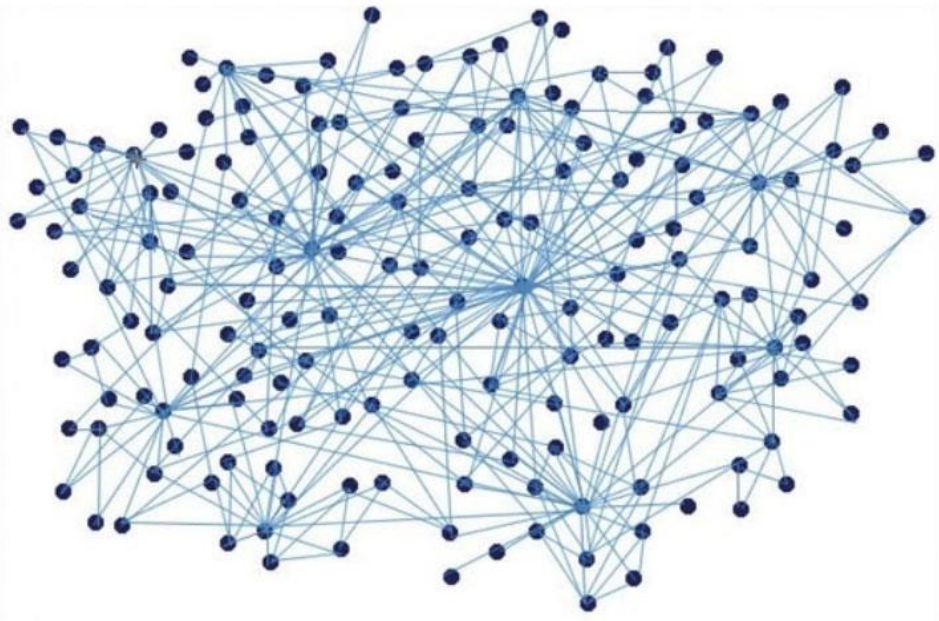




Ejemplo: datos de vuelos

- https://openflights.org/data.html





Parte II: Teoría de grafos

Paulo Villegas

Centralidad

Centralidad

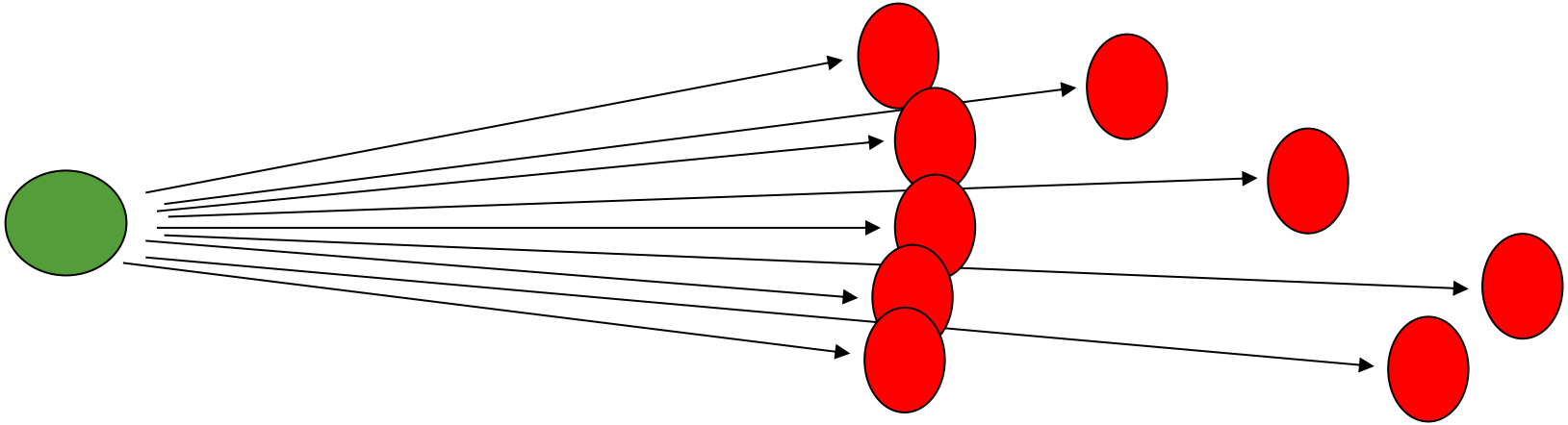
- La centralidad es una propiedad de los nodos de una red
- Trata de medir qué nodos son “centrales” (importantes, críticos) a una red
- Existen diferentes formas de definir lo que es un “centro”
- Por eso hay diferentes métricas de centralidad
  - de grado (*degree centrality*)
  - de proximidad (*closeness*)
  - de intermediación (*betweenness*)
  - autocentralidad (autovalores)
  - *power centrality*

El valor de una red

¿Cuánto vale una red?

- Sarnoff's law:

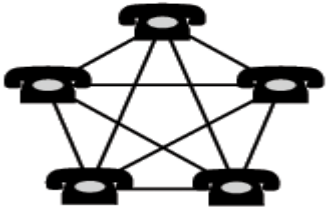
  El valor de una red de difusión (*broadcast*) es proporcional al número de usuarios

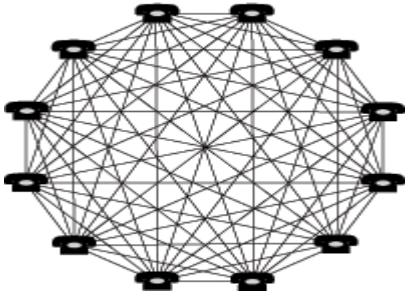


¿Cuánto vale una red?

- **Metcalfe's law:**

  The value of a communication network is proportional to the square of the number of users





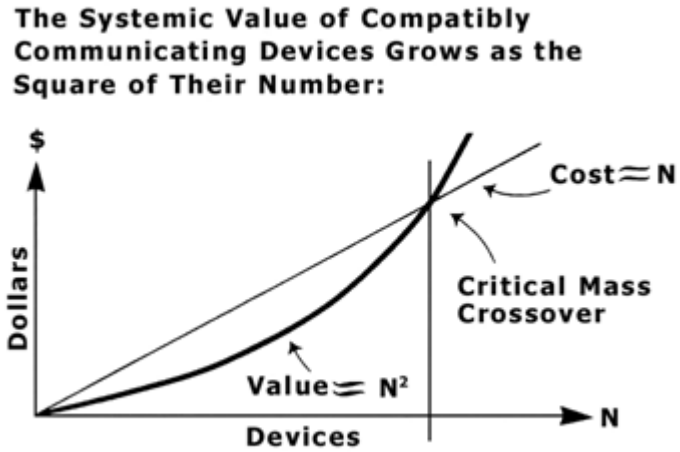
- Why?

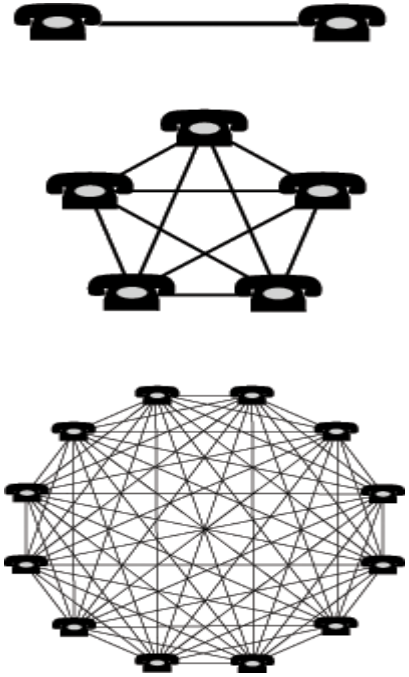
  In a general communication network with n members, there are $n(n - 1)/2$ connections that can be made between pairs of participants

  If all those connections are equally valuable, the total value of the network is proportional to $n(n - 1)/2 \approx n^2$

¿Cuánto vale una red?

- **Metcalfe's law:**





¿Cuánto vale una red?

- **Reed's law:**

  The utility of large networks, particularly social networks, can scale exponentially with the size of the network.

- Why?
  - In a communication network as flexible as the Internet, in addition to linking pairs of members, one can form groups.
  - With n participants, there are 2^n possible groups, and if they are all equally valuable, the value of the network grows like 2^n

¿Cuánto vale una red?

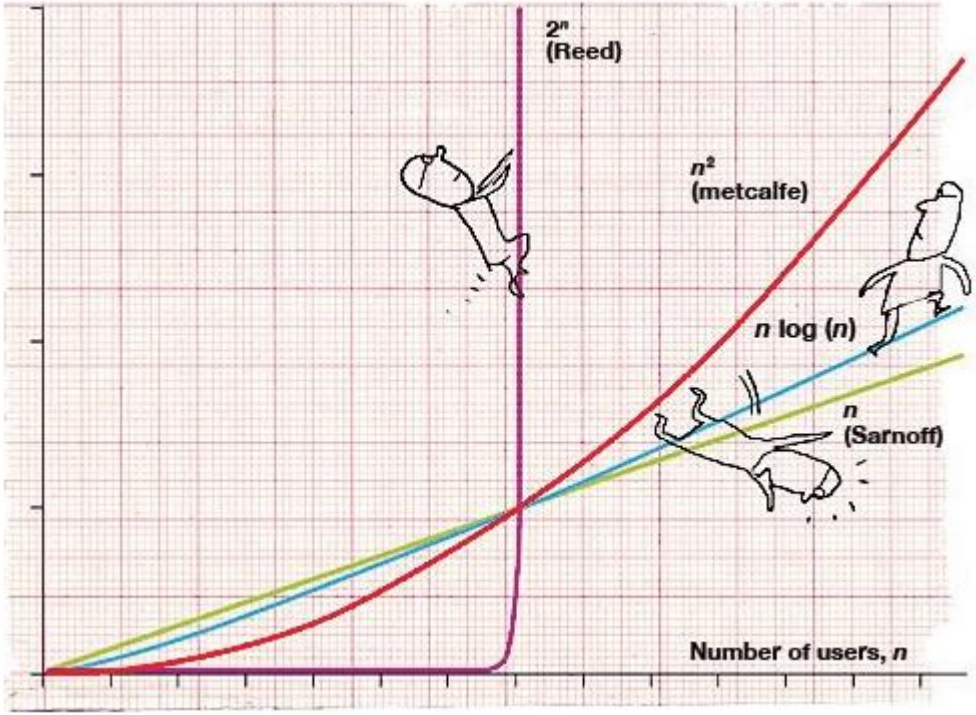
- **Crítica a Metcalfe & Reed**

  Connections are not used with the same intensity (and most are not used at all in large networks, such as the Internet), so assigning equal value to them is not justified

  [Odlyzko & Tilly, 2005]

¿Cuánto vale una red?

- Odlyzko & Tilly proponen *n log(n)* como valoración de una red



Small worlds, power laws & long tails

Small-world network

- Una **small-world network** es un tipo de grafo en el que:
  - La mayoría de los nodos no son vecinos de los demás
  - Pero la mayoría de los nodos pueden alcanzarse desde cualquier otro en un pequeño número de saltos
- En otras palabras
  - La conectividad local es pequeña → la red no es muy densa
  - Pero hay caminos eficientes que conectan cualquier par de nodos arbitrarios → la red está muy conectada
- Ejemplos típicos
  - Redes sociales
  - Internet

Power laws, long tails, etc

- La distribución gaussiana es la más usada en ciencias naturales y sociales
- Sin embargo muchos fenómenos observables no siguen una distribución gaussiana
- Entre ellos hay una clase de eventos que se puede caracterizar bien mediante una ley de potencia (**Power Law**)

$$y \approx x^{-a}$$

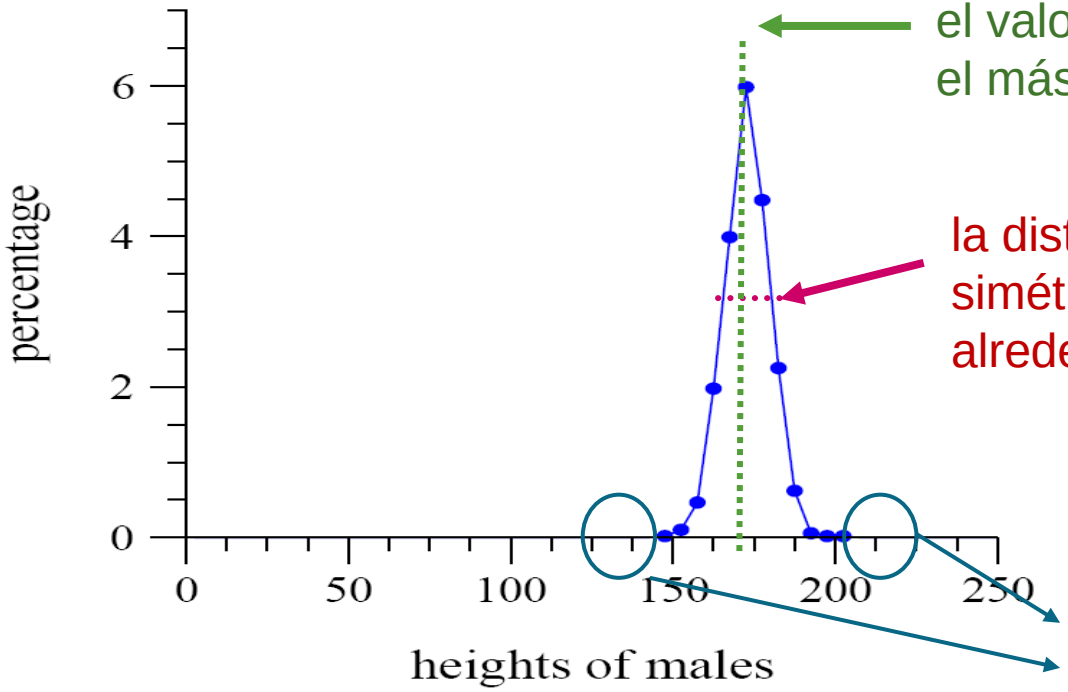
Power laws, long tails, etc

- Las leyes de potencia reciben muchos nombres:
  - Heavy tailed distribution
  - Pareto law
  - Zipf's law
  - 80/20 rule

$$y \approx x^{-a}$$

- En una ley de potencia, *los eventos grandes son raros, <u>pero no nulos</u>, y los eventos pequeños son muy frecuentes*

Distribución normal (gaussiana) de alturas de personas

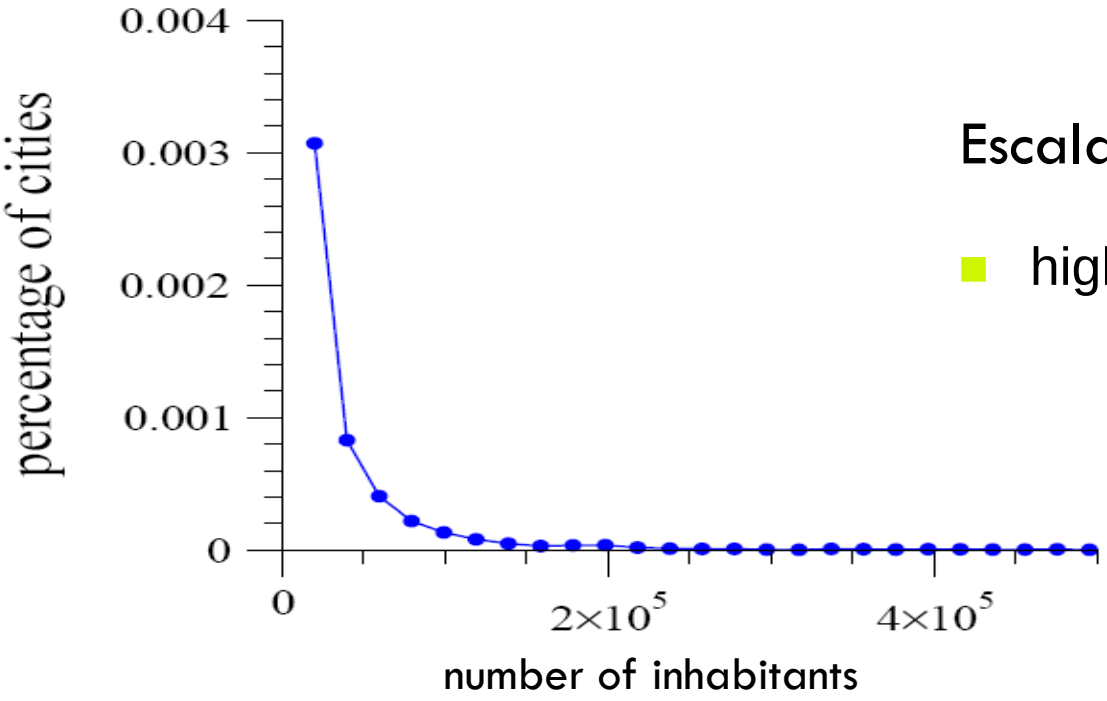


el valor medio suele ser el más típico

la distribución es aprox. simétrica alrededor del valor medio

no hay valores muy alejados del centro

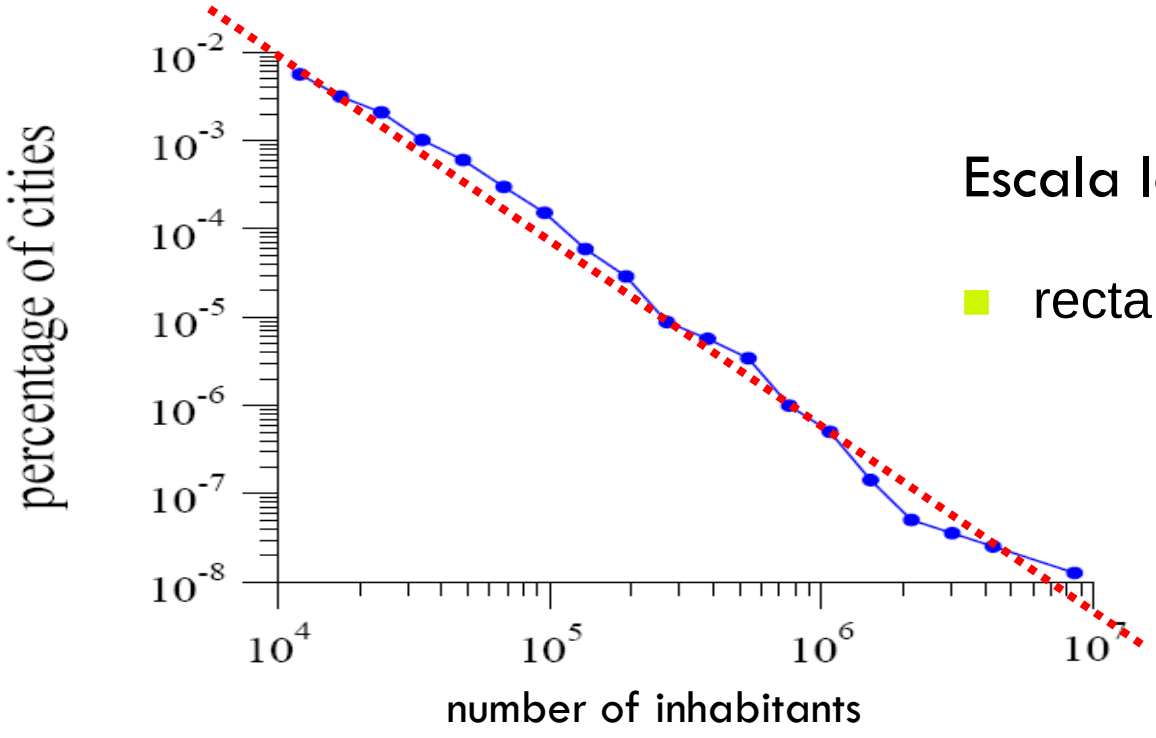
Power-law distribution



Escala lineal

- high skew (asymmetry)

Power-law distribution



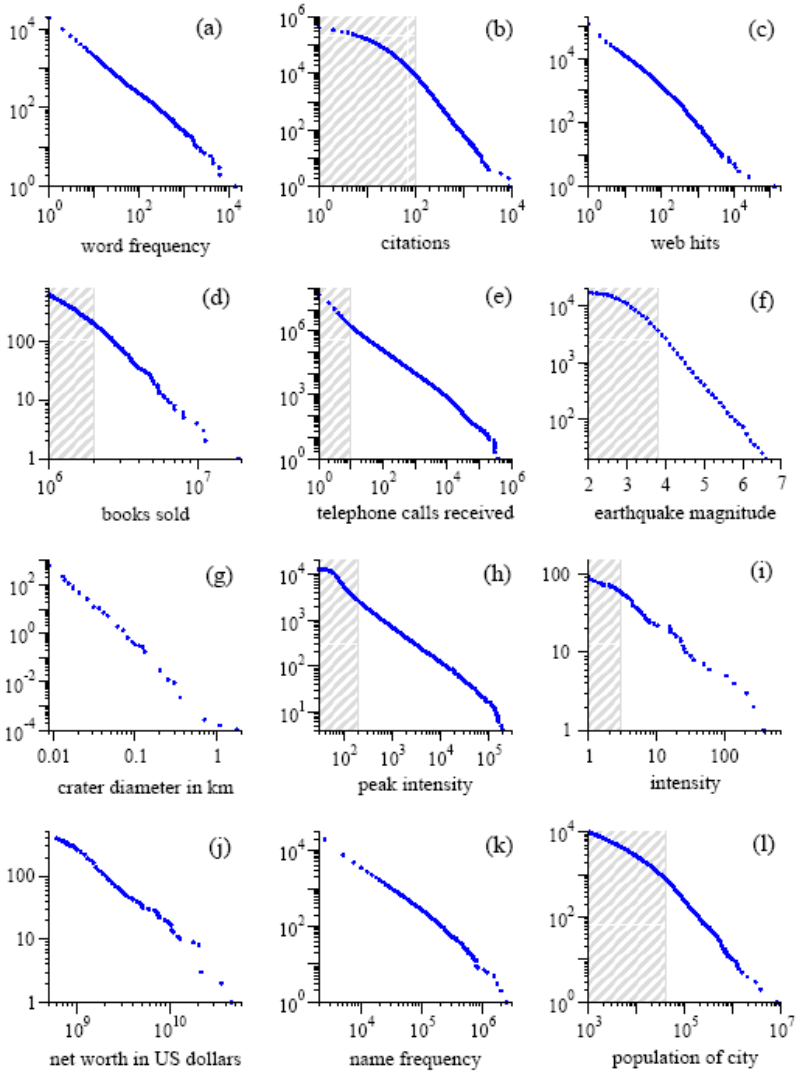
Escala logarítmica

- recta en una gráfica log-log

Algunas distribuciones de leyes de potencia observadas

	quantity	minimum $x_{\min}$	exponent α
(a)	frequency of use of words	1	2.20(1)
(b)	number of citations to papers	100	3.04(2)
(c)	number of hits on web sites	1	2.40(1)
(d)	copies of books sold in the US	2 000 000	3.51(16)
(e)	telephone calls received	10	2.22(1)
(f)	magnitude of earthquakes	3.8	3.04(4)
(g)	diameter of moon craters	0.01	3.14(5)
(h)	intensity of solar flares	200	1.83(2)
(i)	intensity of wars	3	1.80(9)
(j)	net worth of Americans	$600m	2.09(4)
(k)	frequency of family names	10 000	1.94(1)
(l)	population of US cities	40 000	2.30(5)

TABLE I Parameters for the distributions shown in Fig. 4. The labels on the left refer to the panels in the figure. Exponent values were calculated using the maximum likelihood method of Eq. (5) and Appendix B, except for the moon craters (g), for which only cumulative data were available. For this case the exponent quoted is from a simple least-squares fit and should be treated with caution. Numbers in parentheses give the standard error on the trailing figures.



Scale-free networks (redes libres de escala)

- Las redes con distribuciones de grado en forma de ley de potencia se llaman frecuentemente **scale-free networks**
  - La propiedad de independencia de escala se refiere **solo** a la distribución de grados

Saul Steinberg, *View of the World from 9th Avenue*, 1976.

Scale-free networks (redes libres de escala)

- Que los grados sean independientes de la escala equivalen a decir que existen valores de grado para todas las escalas
  - Muchos nodos tienen grados muy bajos (muy poco conectados)
  - Pero algunos tienen grados muy altos (hubs)

Saul Steinberg, *View of the World from 9th Avenue*, 1976.

Scale-free networks (redes libres de escala)

- ¿Por qué existen los hubs?
  - El efecto Mateo – más técnicamente, **asignación preferencial** (*preferential attachment)*
  - Los nodos muy bien conectados atraen nuevos enlaces

Saul Steinberg, *View of the World from 9th Avenue*, 1976.

Scale-free networks (redes libres de escala)

- ¿Cómo se comportan las redes libres de escala?
  - Son robustas frente a ataques aleatorios (eliminar enlaces de forma arbitraria)
  - Son sensibles a ataques dirigidos: eliminar los nodos muy conectados (hubs) puede llegar a hacer perder conectividad

Saul Steinberg, *View of the World from 9th Avenue*, 1976.

Dinámica de redes

Cascadas de información

- Una cascada de información es un procese en el cual una acción o idea adquiere una gran difusión o adopción debido a la influencia de otros
- Las redes sociales juegan un rol fundamental en la difusión de información o influencias
- Viralidad o marketing viral (*boca a boca*)
  - Ideas que adquieren popularidad amplia de golpe
- La difusión de innovaciones es un proceso estudiado desde hace tiempo en sociología:
  - Hybrid corn (Ryan and Gross, 1943)
  - Prescription drugs (Coleman et al. 1957)

Modelos de propagación

- Una vez definida una topología de un grafo (nodos y enlaces), se pueden definir modelos matemáticos que expresan la difusión a través del grafo
- La difusión se realiza de un nodo a otro por los enlaces disponibles, típicamente en iteraciones
- Son útiles en sociología (difusión de rumores), economía (difusión de innovación), epidemiología (propagación de enfermedades)

(a) Modelos compartimentales

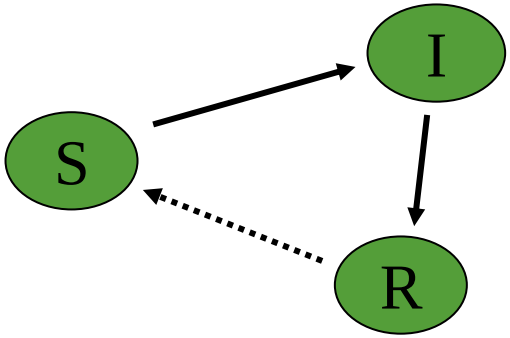
Modelos compartimentales (compartment models)

- Un conjunto de modelos para la propagación de procesos infecciosos
- La población se divide en **compartimentos** con etiquetas
- Las personas progresan entre compartimentos, de acuerdo con ciertas reglas o ecuaciones

Modelo SIR

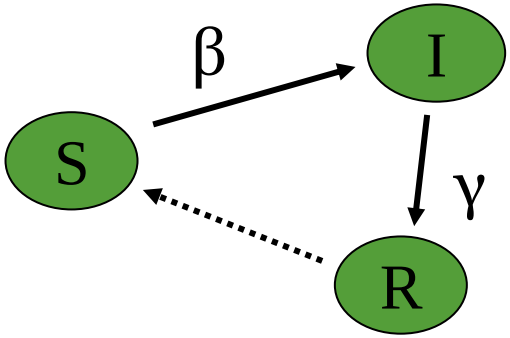
Cada persona puede estar en uno de 3 estados

- Susceptible: sano, pero no inmune
- Infectado: contiene la enfermedad y puede propagarla
- Recuperado: (Removed/Immune/Dead) tuvo la enfermedad pero ya no está activa



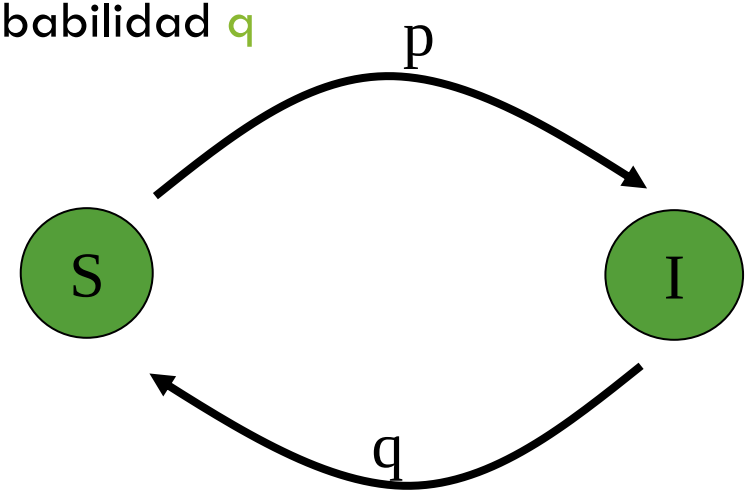
Modelo SIR

- Tasa de infección β: probabilidad de ser contagiado por un vecino infectado, en cada unidad de tiempo
- Tasa de inmunización γ: probabilidad de recuperación de un nodo por unidad de tiempo



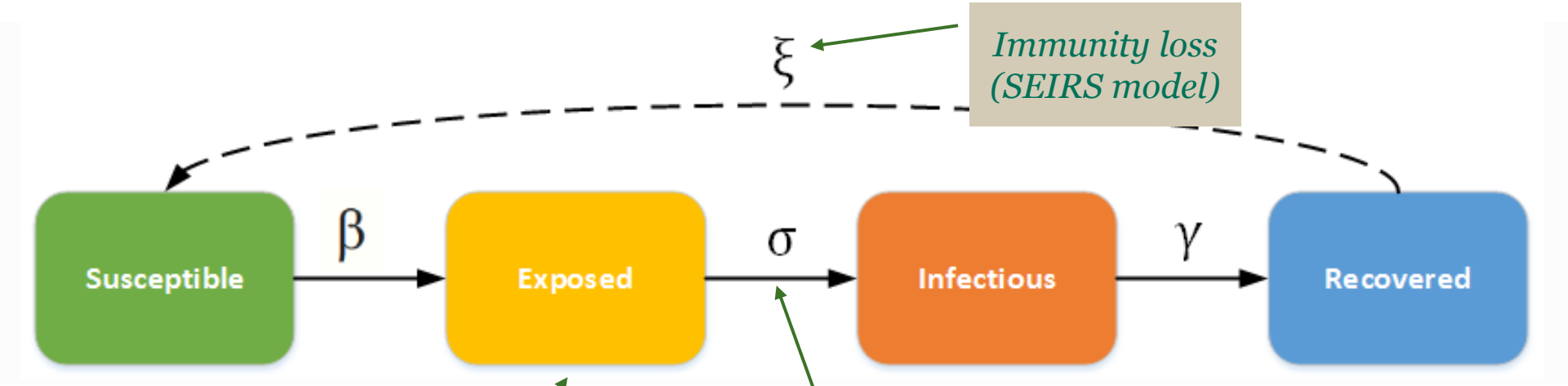
Modelo SIS

- Susceptible-Infectado-Susceptible:
  - Una persona puede estar sana (susceptible) o infectada. No hay inmunidad
  - Un nodo sano con un vecino infectado se infecta con probabilidad p
  - Un nodo infectado se recupera con probabilidad q
  - spreading rate r=p/q



SEIR/SEIRS

Susceptible (S), Exposed (E), Infected (I), Removed (R)

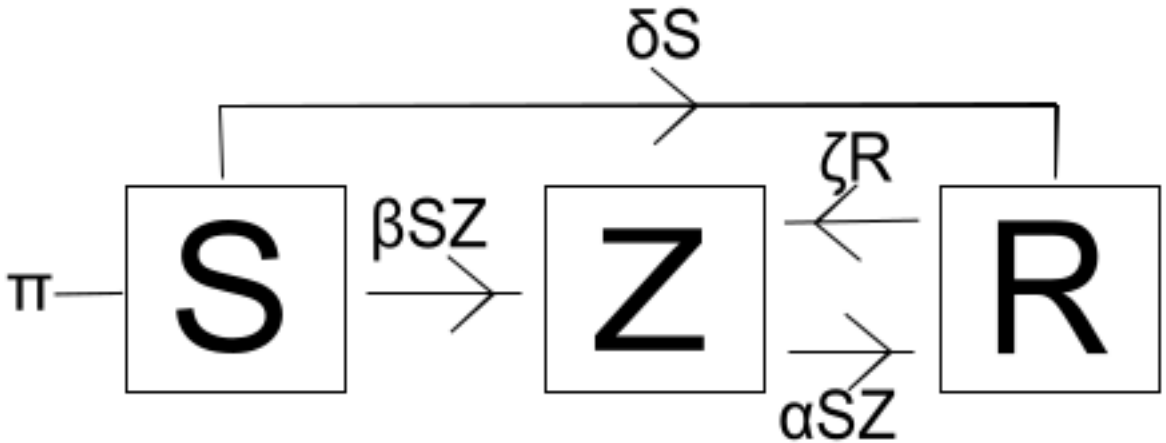


Infected, not yet contagious

incubation rate

The SZR model

Philip Munz, Ioan Hudea, Joe Imad, Robert J. Smith?, *When Zombies Attack!: Mathematical Modelling Of An Outbreak Of Zombie Infection*, 2009



susceptible

undead

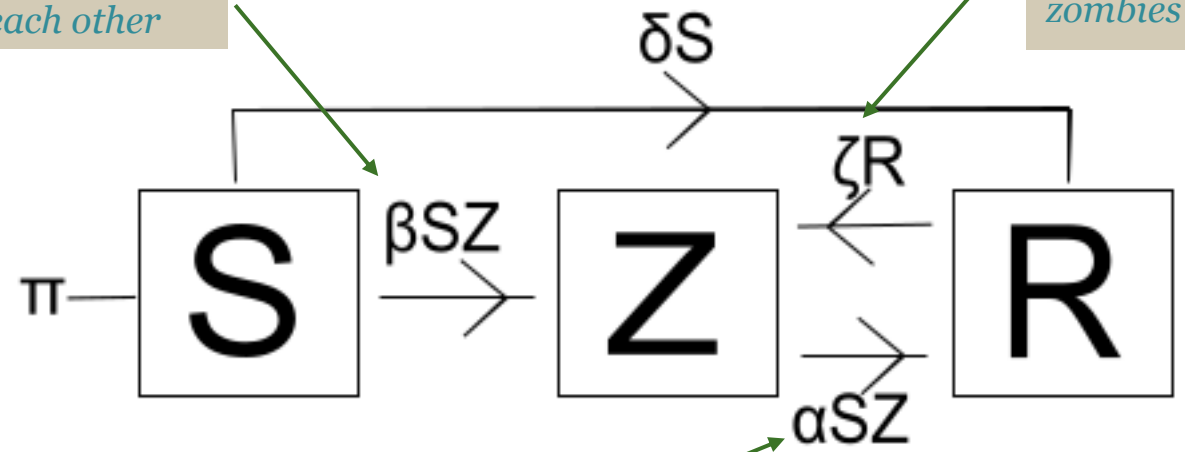
dead

The SZR model

Philip Munz, Ioan Hudea, Joe Imad, Robert J. Smith?, *When Zombies Attack!: Mathematical Modelling Of An Outbreak Of Zombie Infection*, 2009

β - *the probability that a susceptible becomes a zombie when the two encounter each other*

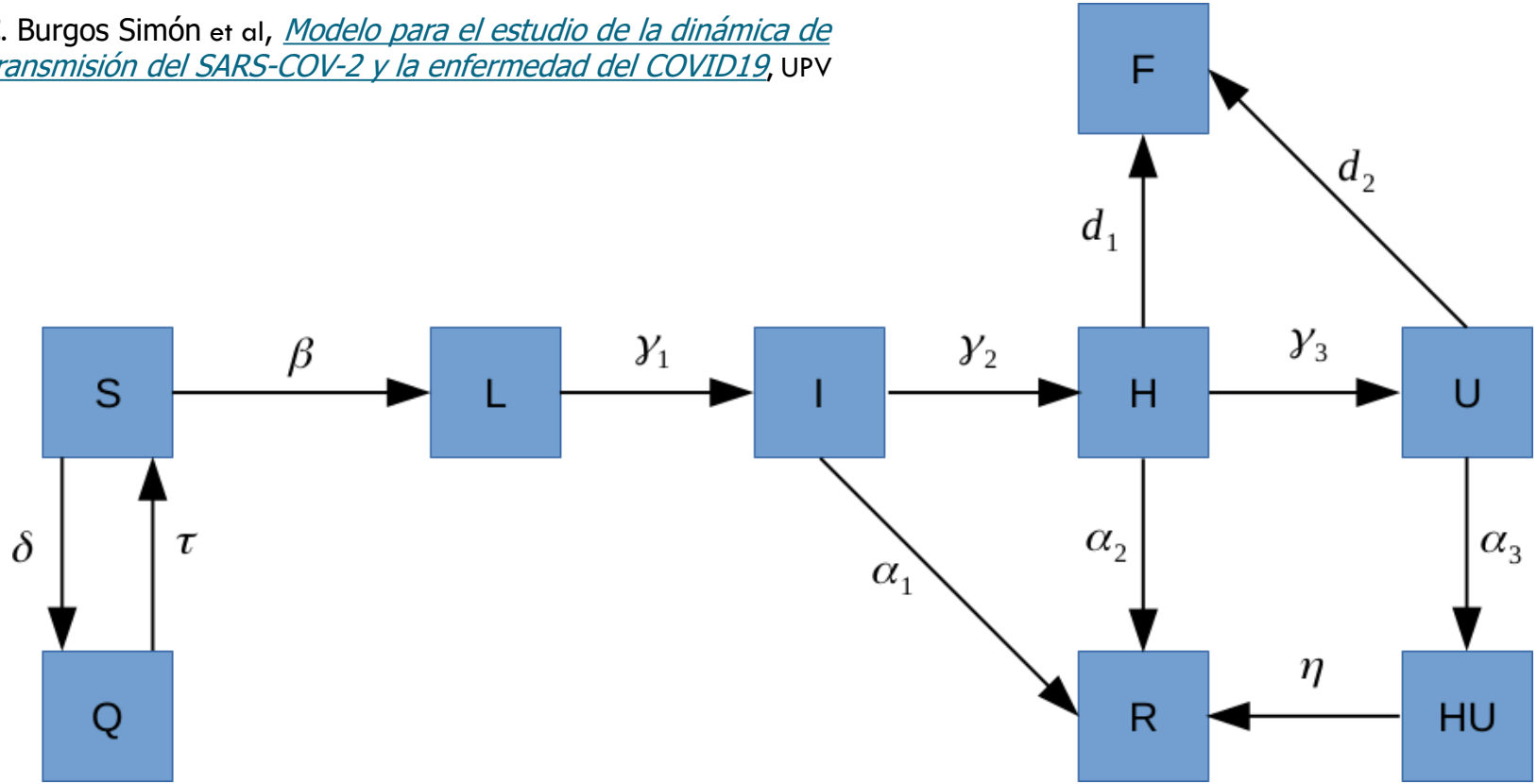
ζ - *the rate at which humans in the removed class resurrect to become zombies*



α - *the rate at which zombies are slain, by removing their heads or destroying their brains*

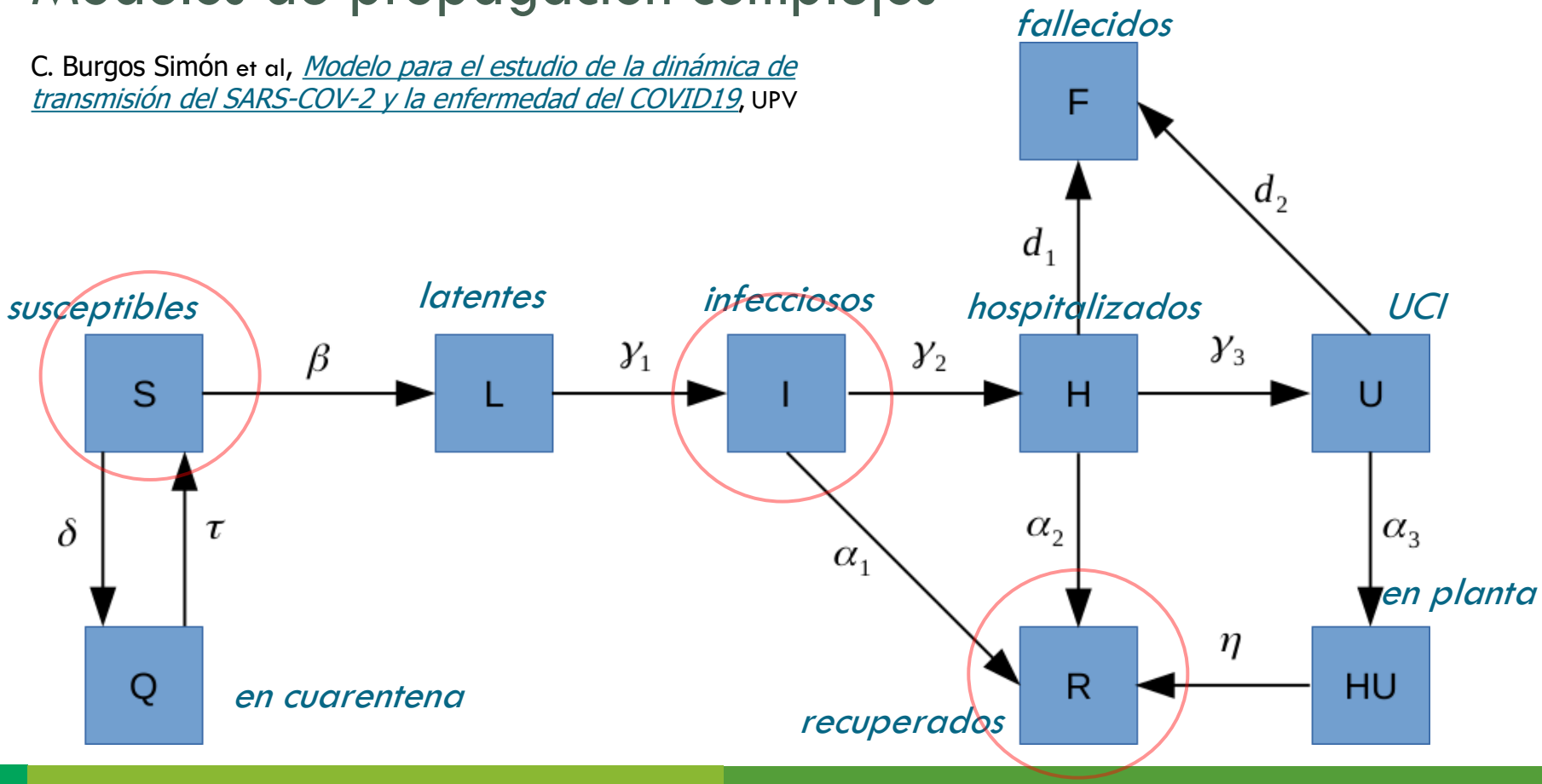
Modelos de propagación complejos

C. Burgos Simón et al, *Modelo para el estudio de la dinámica de transmisión del SARS-COV-2 y la enfermedad del COVID19*, UPV



Modelos de propagación complejos

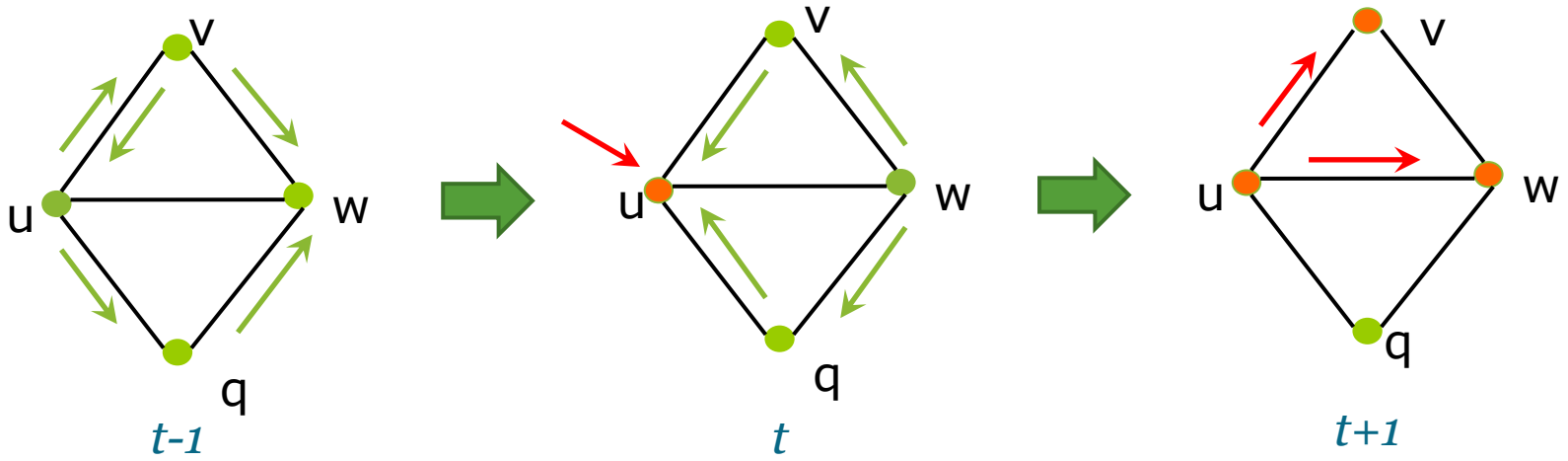
C. Burgos Simón et al, *Modelo para el estudio de la dinámica de transmisión del SARS-COV-2 y la enfermedad del COVID19*, UPV



(b) Propagación de modelos compartimentales *en grafos*

Propagación en un modelo SIR

- El tiempo transcurre en pasos discretos
- Un nodo que fue infectado en el instante *t*, en el instante *t+1* infecta a sus vecinos con probabilidad β
- Al cabo de un tiempo (dado por $1/\gamma$) el nodo pasa a estado R



Percolación

- La percolación es un concepto físico (difusión de líquidos)
- En teoría de redes modelar la creación de un grafo de forma aleatoria, mediante procesos de difusión (se ocupan nodos con probabilidad p)
- A partir de un umbral p_c (*percolation threshold*) la red forma un **componente gigante conectado**
- Hay dos variantes:
  - Site Percolation: Cada nodo de la red se asigna aleatoriamente como ocupado o no-ocupado. con probabilidad p. Buscamos medir el tamaño del máximo componente conectado de vértices ocupados
  - Bond Percolation: Cada enlace de la red se asigna aleatoriamente como ocupado o no-ocupado., con probabilidad p. Buscamos medir el tamaño del máximo componente conectado de enlaces ocupados

Modelo SIR y percolación

- Se puede probar que la propagación en el modelo SIR corresponde a un problema de bond-percolation en la red
- Si la red es libre de escala no hay umbral de percolación
  - La existencia de los hubs permite que la percolación siempre sea teóricamente posible
  - La propagación SIR produce percolación en la red para cualquier tasa de infección mayor que cero, vía propagación jerárquica desde nodos hub hasta nodos con grado bajo

Umbral epidémico

- El umbral epidémico en un modelo SIS es un valor r_c tal que cuando $r < r_c$ la epidemia se desvanece, mientras que con $r > r_c$ se difunde
  - Para grafos homogéneos, si el grado medio es $\langle k \rangle$

$$r_c = \frac{1}{\langle k \rangle}$$

  - Para redes libres de escala

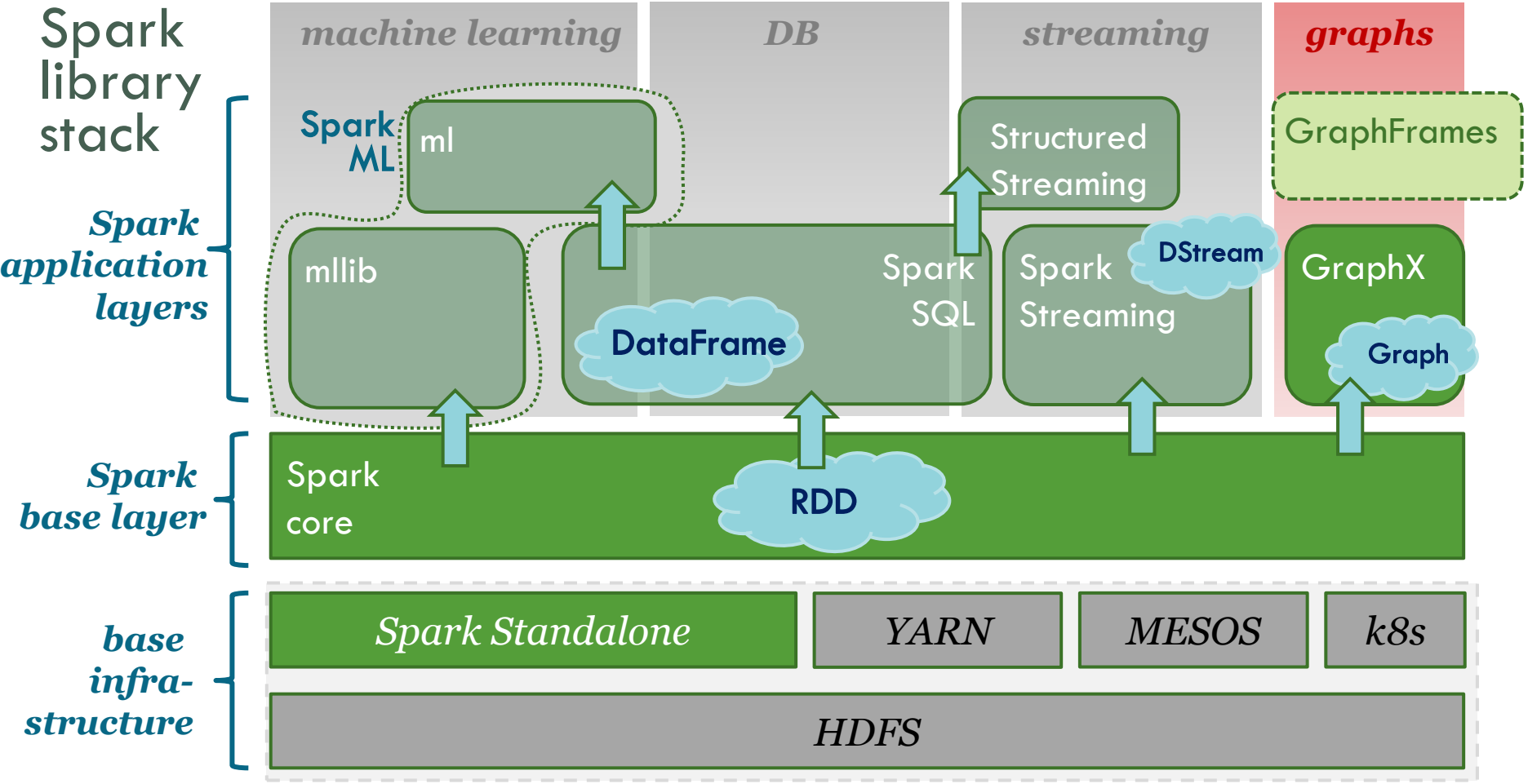
$$r_c = \frac{\langle k \rangle}{\langle k^2 \rangle}$$

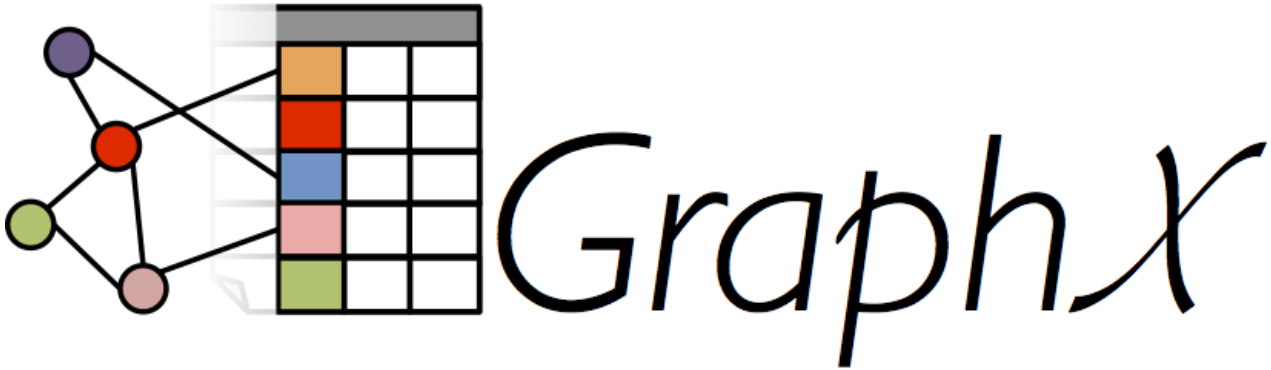
- En una red libre de escala
  - el grado sigue una ley de potencia, $P(k) = k^{-\gamma}$
  - Si $0 < \gamma \leq 3$ la varianza $\langle k^2 \rangle$ tiende a infinito
  - luego el umbral epidémico es cero: la enfermedad siempre se propaga

Parte III: Grafos y Spark

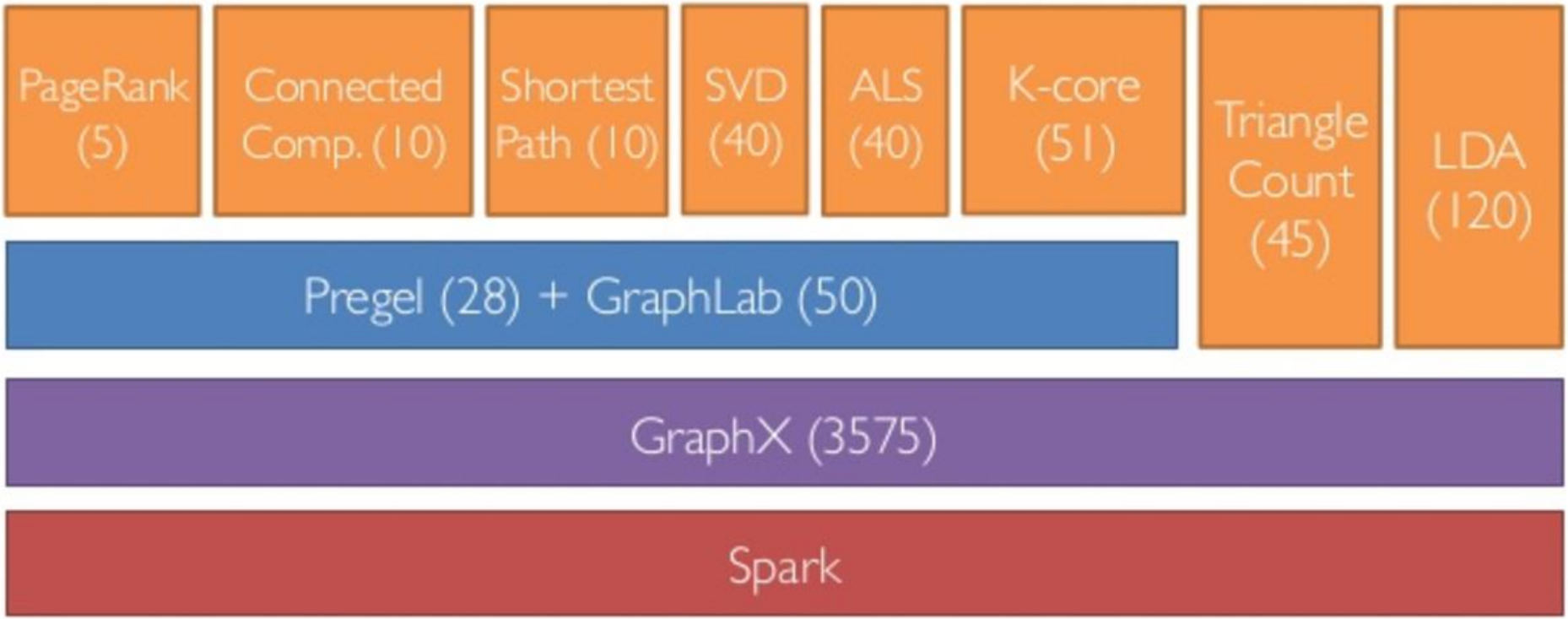
Paulo Villegas

Spark library stack




GraphX

Librería de grafos en Spark: Graphx



Estado de GraphX

- Solo tiene API de Scala
- Está basado en RDDs
- No tiene una gran actividad de desarrollo

- Credits

Core, PySpark and Spark SQ

- **Major features**
  - **Spark on Kubernetes**: [SPARK-1827
    a cluster managed by kubernetes. Not
    configurations, container images and
  - **Vectorized ORC Reader**: [SPARK-16
    throughput through vectorization (2-5x
  - **Spark History Server V2**: [SPARK-18
    scale applications with a more efficien
  - **Data source API V2**: [SPARK-15689][
    API attempts to address several limitati
    maintain, and extensible external data
    should be expected.
  - **PySpark Performance Enhancemer**
    interoperability by fast data serializatio
- **Performance and stability**
  - [SPARK-21975] Histogram support in c
  - [SPARK-20331] Better support for pred
  - [SPARK-19112] Support for ZStandard
  - [SPARK-21113] Support for read ahea
  - [SPARK-22510][SPARK-22692][SPARK
    limit on the Java method and Java con
  - [SPARK-23207] Fixed a long standing
    incorrect answers in certain surgical ca
  - [SPARK-22062][SPARK-17788][SPARK
  - [SPARK-22489][SPARK-22916][SPARK
    Enhancements in rule-based optimizer
- **Other notable changes**
  - [SPARK-20236] Support Hive style dyn
  - [SPARK-4131] Support `INSERT OVERW`
  - [SPARK-19285][SPARK-22945][SPARK
  - [SPARK-20463][SPARK-19951][SPARK
    [SPARK-17642][SPARK-22475][SPARK
  - [SPARK-20746] More comprehensive S
  - [SPARK-21485] Spark SQL documenta
  - [SPARK-19810] Remove support for Sc
  - [SPARK-22324] Upgrade Arrow to `0.8`

Programming guides: Spark RDD Programming Gu

Programming guides: Spark RDD Programming Guide and

Structured Streaming

- **Continuous Processing**
  - A new execution engine that can execute stre
    line of user code. To learn more see the progr
- **Stream-Stream Joins**
  - Ability to join two streams of data, buffering ro
    against event time columns to bound the amo
- **Streaming API V2**
  - An experimental API for plugging in new sour
    this API is still undergoing active developmen

Programming guide: Structured Streaming Programming G

MLlib

- **Highlights**
  - ML Prediction now works with Structured Stre
- **New/Improved APIs**
  - [SPARK-21866]: Built-in support for reading im
  - [SPARK-19634]: DataFrame functions for des
  - [SPARK-14516]: `ClusteringEvaluator` for
    silhouette metrics (Scala/Java/Python)
  - [SPARK-3181]: Robust linear regression with
  - [SPARK-13969]: `FeatureHasher` transformer
  - Multiple column support for several feature tra
    - [SPARK-13030]: `OneHotEncoderEsti`
    - [SPARK-22397]: `QuantileDiscretiz`
    - [SPARK-20542]: `Bucketizer` (Scala/Ja
  - [SPARK-21633] and SPARK-21542]: Improved
- **New Features**
  - [SPARK-21087]: `CrossValidator` and `Train`
    you to inspect or save all fitted models.
  - [SPARK-19357]: Meta-algorithms `CrossVali`
    fitting multiple sub-models in parallel Spark jo
  - [SPARK-17139]: Model summary for multinom
  - [SPARK-18710]: Add offset in GLM
  - [SPARK-20199]: Added `featureSubsetStra`
    features can significantly improve training spe
- **Other Notable Changes**
  - [SPARK-22156] Fixed `Word2Vec` learning rate
  - [SPARK-22289] Add `JSON` support for Matrix parame
    `LogisticRegressionModel` when using bounds o
  - [SPARK-22700] `Bucketizer.transform` incorrectly
    "skip," `Bucketizer` would drop a row with a valid va
  - [SPARK-22446] Catalyst optimizer sometimes cause
    when `handleInvalid` was set to "error." This could
    after invalid rows had already been filtered from the i
  - [SPARK-21681] Fixed an edge case bug in multinom
    features had zero variance.
  - Major optimizations:
    - [SPARK-22707] Reduced memory consumptio
    - [SPARK-22949] Reduced memory consumptio
    - [SPARK-21690] `Imputer` should train using a
    - [SPARK-14371] `OnlineLDAOptimizer` avoid

Programming guide: Machine Learning Library (MLlib) Guide.

SparkR

The main focus of SparkR in the 2.3.0 release was towards impro
around existing APIs:

- **Major features**
  - Improved function parity between SQL and R
  - [SPARK-22933]: Structured Streaming APIs for `withW`
  - [SPARK-21266]: SparkR UDF with DDL-formatted sc
  - [SPARK-20726][SPARK-22924][SPARK-22843] Seve
  - [SPARK-15767][SPARK-21622][SPARK-20917][SPAR

Programming guide: SparkR (R on Spark).

GraphX

- **Optimizations**
  - [SPARK-5484] Pregel now checkpoints periodically t
  - [SPARK-21491] Small performance improvement in s

Programming guide: GraphX Programming Guide.

Deprecations

- **Python**
  - [SPARK-23122] Deprecate `register*` for UDFs in S
- **MLlib**
  - [SPARK-13030] `OneHotEncoder` has been deprecat

Changes of behavior

- **SparkSQL**
  - [SPARK-22036]: By default arithmetic operations between d
    possible (instead of returning `NULL` in the prior versions)
  - [SPARK-22937]: When all inputs are binary, SQL `elt()` retu
    versions, it always returns as a string despite of input types.
  - [SPARK-22895]: The Join/Filter's deterministic predicates th
    down/through the child operators, if possible. In the prior ve
  - [SPARK-22771]: When all inputs are binary, `functions.co`
    string. In the prior versions, it always returns as a string des
  - [SPARK-22489]: When either of the join sides is broadcasta
    a broadcast hint.
  - [SPARK-22165]: Partition column inference previously foun
    previously it ended up with `double` type as the common typ
    type for such conflicts. For details, see the migration guide.
  - [SPARK-22100]: The `percentile_approx` function previou
    results. Now it supports `date` type, `timestamp` type and nu
    the same as the input type, which is more reasonable for pe
  - [SPARK-21610]: the queries from raw JSON/CSV files are d
    corrupt record column (named `_corrupt_record` by defau
    the same query.
  - [SPARK-23421]: Since Spark 2.2.1 and 2.3.0, the schema is
    columns that exist in both partition schema and data schem
    When reading the table, Spark respects the partition values
    data source files. In 2.2.0 and 2.1.x release, the inferred sch
    the result set is empty).
- **PySpark**
  - [SPARK-19732]: `na.fill()` or `fillna` also accepts boolea
    PySpark just ignores it and returns the original Dataset/Data
  - [SPARK-22395]: Pandas `0.19.2` or upper is required for us
    `createDataFrame` from Pandas DataFrame, etc.
  - [SPARK-22395]: The behavior of timestamp values for Pand
    timezone, which is ignored in the prior versions.
  - [SPARK-23328]: `df.replace` does not allow to omit `value`
    omitted in the other cases and had `None` by default, which is
- **MLlib**
  - **Breaking API Changes**: The class and trait hierarchy for lo
    better accommodate the addition of the multi-class summar
    `LogisticRegressionTrainingSummary` to a `BinaryLog`
    the `model.binarySummary` method. See [SPARK-17139] f
    affect the Python summary method, which will still work corr

- Credits

Core, PySpark and Spark SQ

- **Major features**
  - **Spark on Kubernetes**: [SPARK-1827
    a cluster managed by kubernetes. Not
    configurations, container images and
  - **Vectorized ORC Reader**: [SPARK-16
    throughput through vectorization (2-5x
  - **Spark History Server V2**: [SPARK-18
    scale applications with a more efficien
  - **Data source API V2**: [SPARK-15689][
    API attempts to address several limitati
    maintain, and extensible external data
    should be expected.
  - **PySpark Performance Enhancemer**
    interoperability by fast data serializatio
- **Performance and stability**
  - [SPARK-21975] Histogram support in c
  - [SPARK-20331] Better support for pred
  - [SPARK-19112] Support for ZStandard
  - [SPARK-21113] Support for read ahea
  - [SPARK-22510][SPARK-22692][SPARK
    limit on the Java method and Java con
  - [SPARK-23207] Fixed a long standing
    incorrect answers in certain surgical ca
  - [SPARK-22062][SPARK-17788][SPARK
  - [SPARK-22489][SPARK-22916][SPARK
    Enhancements in rule-based optimizer
- **Other notable changes**
  - [SPARK-20236] Support Hive style dyn
  - [SPARK-4131] Support `INSERT OVERW`
  - [SPARK-19285][SPARK-22945][SPARK
  - [SPARK-20463][SPARK-19951][SPARK
    [SPARK-17642][SPARK-22475][SPARK
  - [SPARK-20746] More comprehensive
  - [SPARK-21485] Spark SQL documenta
  - [SPARK-19810] Remove support for Sc
  - [SPARK-22324] Upgrade Arrow to `0.8`

Programming guides: Spark RDD Programming Gu

Programming guides: Spark RDD Programming Guide and

Structured Streaming

- **Continuous Processing**
  - A new execution engine that can execute stre
    line of user code. To learn more see the progr
- **Stream-Stream Joins**
  - Ability to join two streams of data, buffering ro
    against event time columns to bound the amo
- **Streaming API V2**
  - An experimental API for plugging in new sour
    this API is still undergoing active developmen

Programming guide: Structured Streaming Programming G

MLlib

- **Highlights**
  - ML Prediction now works with Structured Stre
- **New/Improved APIs**
  - [SPARK-21866]: Built-in support for reading in
  - [SPARK-19634]: DataFrame functions for des
  - [SPARK-14516]: `ClusteringEvaluator` for
    silhouette metrics (Scala/Java/Python)
  - [SPARK-3181]: Robust linear regression with
  - [SPARK-13969]: `FeatureHasher` transformer
  - Multiple column support for several feature tra
    - [SPARK-13030]: `OneHotEncoderEsti`
    - [SPARK-22397]: `QuantileDiscretiz`
    - [SPARK-20542]: `Bucketizer` (Scala/Ja
  - [SPARK-21633] and SPARK-21542]: Improvec
- **New Features**
  - [SPARK-21087]: `CrossValidator` and `Trai`
    you to inspect or save all fitted models.
  - [SPARK-19357]: Meta-algorithms `CrossVali`
    fitting multiple sub-models in parallel Spark jo
  - [SPARK-17139]: Model summary for multinom
  - [SPARK-18710]: Add offset in GLM
  - [SPARK-20199]: Added `featureSubsetStra`
    features can significantly improve training spe
- **Other Notable Changes**
  - [SPARK-22156] Fixed `Word2Vec` learning rate
  - [SPARK-22289] Add `JSON` support for Matrix parame
    `LogisticRegressionModel` when using bounds o
  - [SPARK-22700] `Bucketizer.transform` incorrectly
    "skip," `Bucketizer` would drop a row with a valid va
  - [SPARK-22446] Catalyst optimizer sometimes cause
    when `handleInvalid` was set to "error." This could
    after invalid rows had already been filtered from the i
  - [SPARK-21681] Fixed an edge case bug in multinom
    features had zero variance.
  - Major optimizations:
    - [SPARK-22707] Reduced memory consumptio
    - [SPARK-22949] Reduced memory consumptio
    - [SPARK-21690] `Imputer` should train using a
    - [SPARK-14371] `OnlineLDAOptimizer` avoid

Programming guide: Machine Learning Library (MLlib) Guide.

SparkR

The main focus of SparkR in the 2.3.0 release was towards impro
around existing APIs:

- **Major features**
  - Improved function parity between SQL and R
  - [SPARK-22933]: Structured Streaming APIs for `withl`
  - [SPARK-21266]: SparkR UDF with DDL-formatted sc
  - [SPARK-20726][SPARK-22924][SPARK-22843] Seve
  - [SPARK-15767][SPARK-21622][SPARK-20917][SPAR

Programming guide: SparkR (R on Spark).

GraphX

- **Optimizations**
  - [SPARK-5484] Pregel now checkpoints periodically t
  - [SPARK-21491] Small performance improvement in s

Programming guide: GraphX Programming Guide.

Deprecations

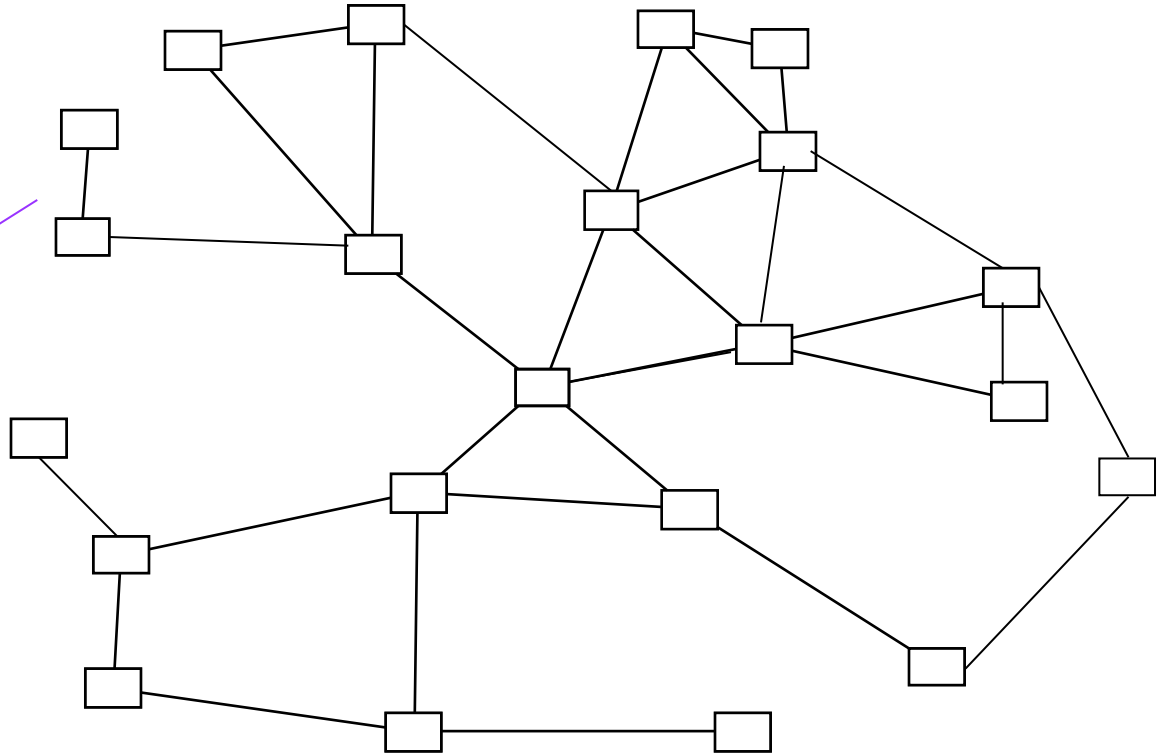
- **Python**
  - [SPARK-23122] Deprecate `register*` for UDFs in S
- **MLlib**
  - [SPARK-13030] `OneHotEncoder` has been deprecat

Changes of behavior

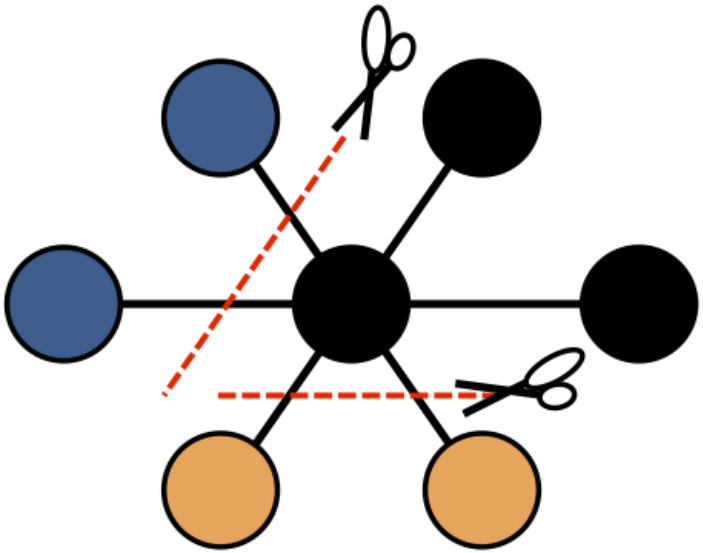
- **SparkSQL**
  - [SPARK-22036]: By default arithmetic operations between d
    possible (instead of returning `NULL` in the prior versions)
  - [SPARK-22937]: When all inputs are binary, SQL `elt()` retu
    versions, it always returns as a string despite of input types.
  - [SPARK-22895]: The Join/Filter's deterministic predicates th
    down/through the child operators, if possible. In the prior ve
  - [SPARK-22771]: When all inputs are binary, `functions.co`
    string. In the prior versions, it always returns as a string des
  - [SPARK-22489]: When either of the join sides is broadcasta
    a broadcast hint.
  - [SPARK-22165]: Partition column inference previously foun
    previously it ended up with `double` type as the common typ
    type for such conflicts. For details, see the migration guide.
  - [SPARK-22100]: The `percentile_approx` function previou
    results. Now it supports `date` type, `timestamp` type and nu
    the same as the input type, which is more reasonable for pe
  - [SPARK-21610]: the queries from raw JSON/CSV files are d
    corrupt record column (named `_corrupt_record` by defau
    the same query.
  - [SPARK-23421]: Since Spark 2.2.1 and 2.3.0, the schema is
    columns that exist in both partition schema and data schem
    When reading the table, Spark respects the partition values
    data source files. In 2.2.0 and 2.1.x release, the inferred sch
    the result set is empty).
- **PySpark**
  - [SPARK-19732]: `na.fill()` or `fillna` also accepts boolea
    PySpark just ignores it and returns the original Dataset/Data
  - [SPARK-22395]: Pandas `0.19.2` or upper is required for us
    `createDataFrame` from Pandas DataFrame, etc.
  - [SPARK-22395]: The behavior of timestamp values for Pand
    timezone, which is ignored in the prior versions.
  - [SPARK-23328]: `df.replace` does not allow to omit `value`
    omitted in the other cases and had `None` by default, which is
- **MLlib**
  - **Breaking API Changes**: The class and trait hierarchy for lo
    better accommodate the addition of the multi-class summar
    `LogisticRegressionTrainingSummary` to a `BinaryLog`
    the `model.binarySummary` method. See [SPARK-17139] f
    affect the Python summary method, which will still work corr

Principios básicos de GraphX

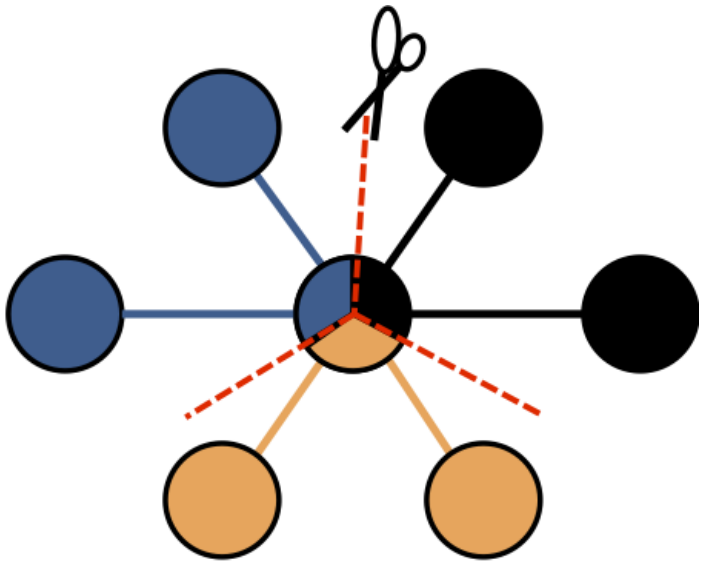
¿cómo rompemos el grafo para repartirlo en distintos nodos?



Principios básicos de GraphX

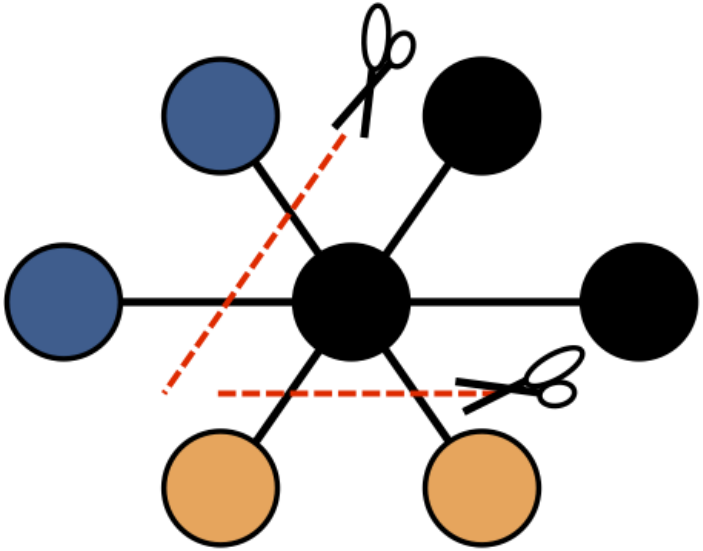


Edge Cut

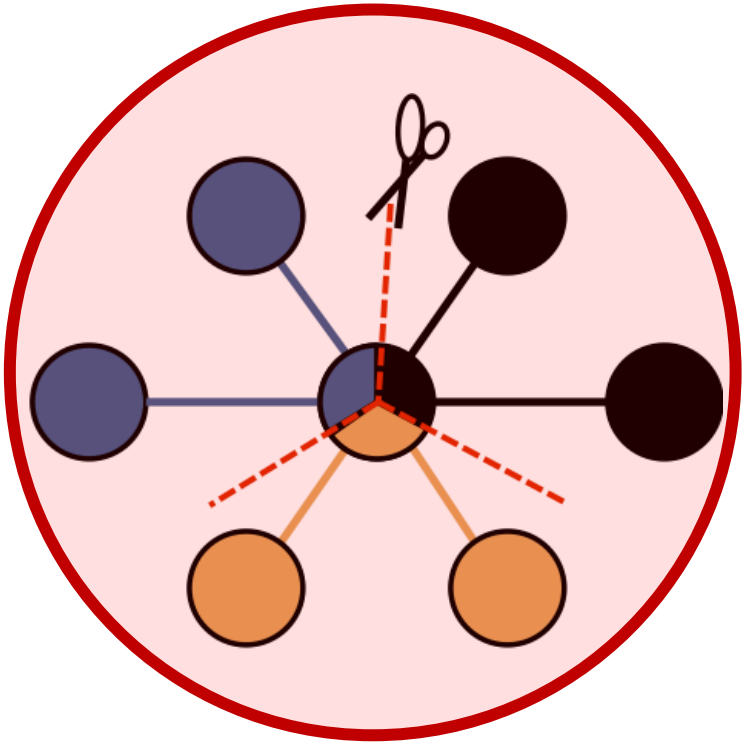


Vertex Cut

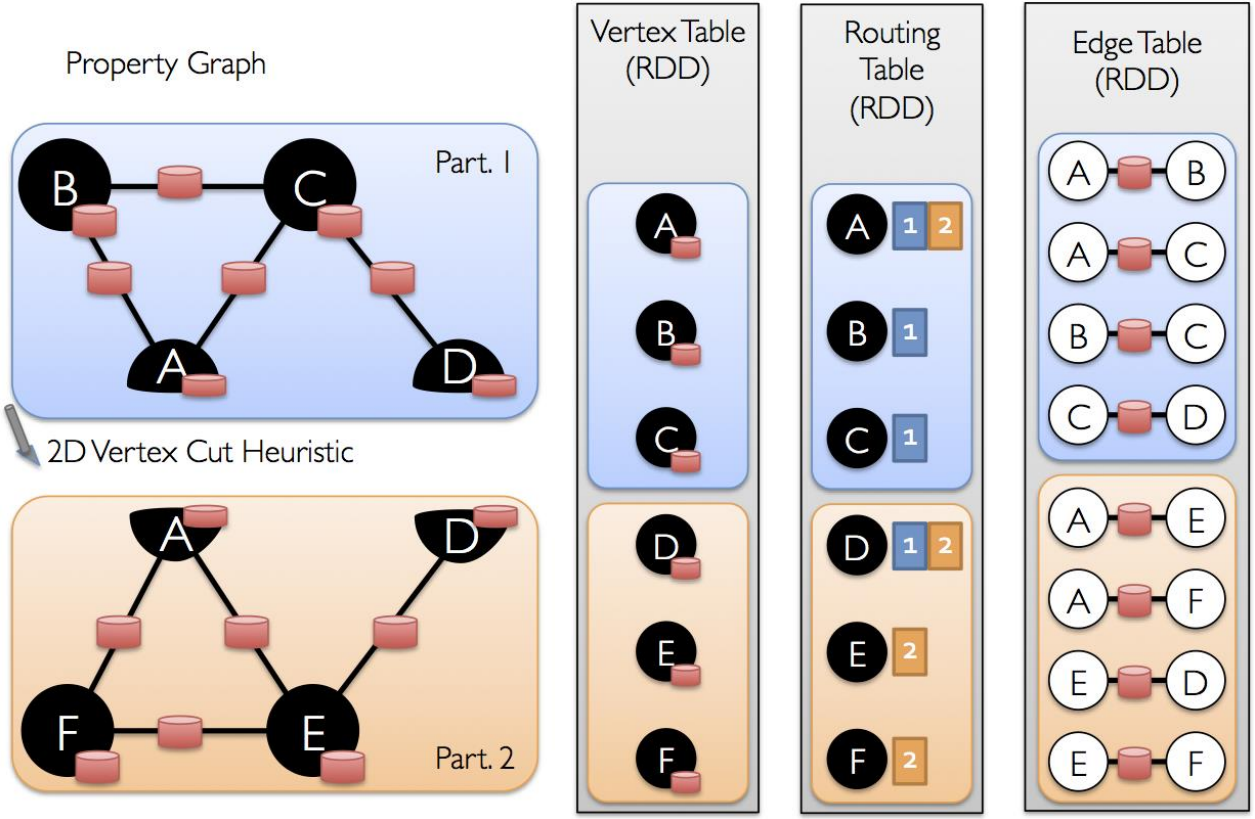
Principios básicos de GraphX



Edge Cut



Vertex Cut


Property Graph
Part. 1
B
C
A
D
2D Vertex Cut Heuristic
A
D
F
E
Part. 2
Vertex Table (RDD)
A
B
C
D
E
F
Routing Table (RDD)
A 1 2
B 1
C 1
D 1 2
E 2
F 2
Edge Table (RDD)
A B
A C
B C
C D
A E
A F
E D
E F

GraphFrames

Grafos + DataFrames ➔ GraphFrames

RDD → GraphX

DataFrames → GraphFrames

GraphFrames

- a package for Apache Spark which provides DataFrame-based Graphs.
- It provides high-level APIs in Scala, Java, and Python.
- It aims to provide both the functionality of GraphX and extended functionality taking advantage of Spark DataFrames

Dave, Ankur, et al. "Graphframes: an integrated api for mixing graph and relational queries." Proceedings of the Fourth International Workshop on Graph Data Management Experiences and Systems. ACM, 2016.

GraphFrames

- No forma parte de la distribución oficial de Spark
  - Se desarrolla en paralelo, en su propio repositorio
  - Se instala por separado, como un add-on
- Proporciona APIs que manipulan grafos mediante DataFrames
  - en Scala, Java y Python
  - hay un paquete adicional para R: R GraphFrames

Instalación de GraphFrames

- En general un paquete adicional de Spark se puede instalar de 4 formas distintas …

Uso de paquetes adicionales en Spark

Tipo de especificación

- Paquete lógico
- Ficheros

Instante de definición

- En ejecución
- En configuración

Opciones de despliegue (paquete)

- 1. Indicar el paquete a cargar al lanzar la aplicación.

  Spark lo descargará automáticamente de Internet (la primera vez) junto con sus dependencias

  ```
  spark-submit --packages graphframes:graphframes:0.8.2-spark3.1-s_2.11 ...
  ```

- 2. Añadir la carga del paquete a la configuración de Spark en `spark-defaults.conf`

  Spark lo descargará de Internet cuando arranque

  ```
  spark.jars.packages=graphframes:graphframes:0.8.2-spark3.1-s_2.11 2
  ```

Opciones de despliegue (ficheros)

- 3. Descargar manualmente el paquete (un `jar`) y añadirlo al lanzar la aplicación.

  Spark lo mandará a todos los ejecutores

  ```
  spark-submit --jars graphframes-0.8.2-spark3.1-s_2.11.jar ...
  ```

- 4. Añadir la carga del fichero a la configuración de Spark en `spark-defaults.conf`

  Spark lo leerá al arrancar y mandará a los ejecutores

  ```
  spark.jars=graphframes-0.8.2-spark3.1-s_2.11.jar
  ```

Cuidado con las dependencias

Despliegue en Python

- El `jar` de graphframes contiene también el código Python para la API de GraphFrames

- Además de desplegar el jar, es necesario ajustar `PYTHONPATH` para que localice las librerías de GraphFrames dentro del `jar`.

Instalación en la máquina virtual

- GraphFrames (versión `0.8.2`) ya viene preconfigurado en `spark-defaults.conf`

- Se activa/desactiva con un provisionado de vagrant

```
vagrant provision --provision-with graphframes
```

- El cambio afecta a todos los procesos (y kernels de Jupyter) lanzados *después* de la provisión

Funcionalidad de GraphFrames

Construcción de un grafo

columna `id`

Los identificadores de nodos id son los usados en las columnas src y dst

```
import graphframes as gf

df_nodes = ..load a DataFrame containing vertices..
df_edges = ..load a DataFrame containing edges..
graph = gf.GraphFrame(df_nodes, df_edges)
```

columnas `src`
`dst`

E/S de grafos

- Un grafo de GraphFrames está soportado internamente por dos DataFrames: vértices y enlaces
- Para poder grabarlo en disco basta volcar a disco esos DataFrames
- Cargando los DataFrames se puede recrear el grafo fácilmente

GraphX a/de GraphFrames

- Existen primitivas para convertir un grafo de GraphX en uno de GraphFrames, y viceversa
- Sólo es posible en Scala

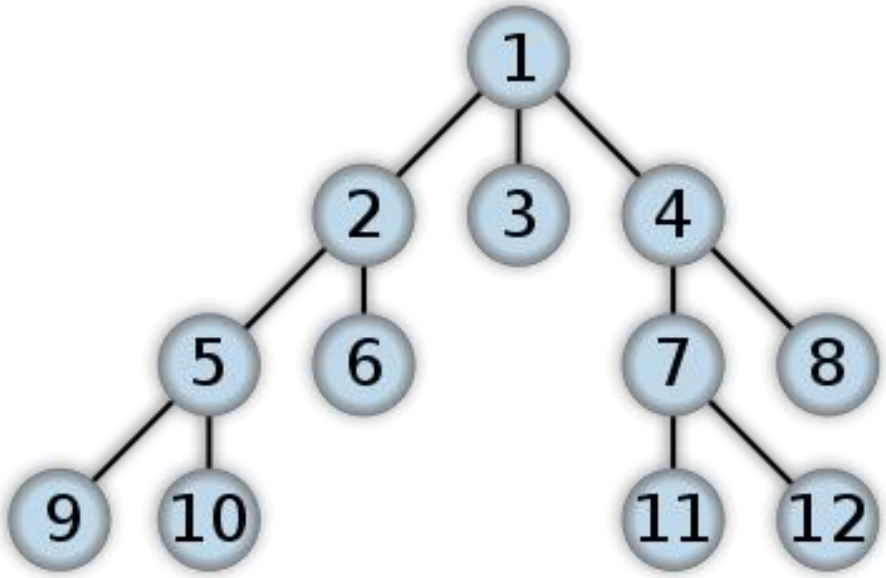
Algoritmos

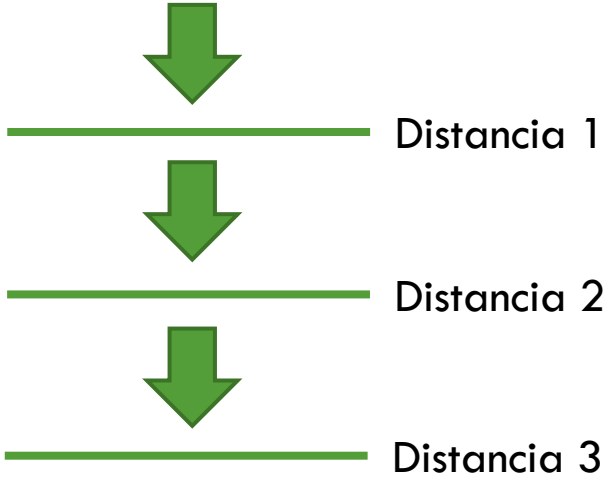
- Grados de los nodos: `inDegrees`, `outDegrees`, `degrees`
- Subgrafos
- Localización de elementos
  - *Motif finding:* `find`
  - Búsquedas: `bfs`
  - Camino más corto: `shortestPath`
- Estructura
  - Conectividad
    - Componentes conectados
    - Componentes fuertemente conectados
  - Detección de comunidades
- Pagerank
- Triangle Count

➔ User Guide

Búsqueda en anchura (BFS)

- Se priorizan las visitas a los nodos por menor distancia al origen de la búsqueda





Búsqueda en anchura con Graph Frames

- Se pueden aplicar filtros y limitar la profundidad

```
paths = g.bfs("name = 'Esther'", "age < 32")
paths.show()


g.bfs("name = 'Esther'", "age < 32",
  edgeFilter="relationship != 'friend'", maxPathLength=3)
```

Búsqueda de patrones (Motif finding)

Se basa en la estructura de tripleta

```
(src) -> [edge] -> (dst)
```

1. Damos unos nombres arbitrarios a los componentes de cada tripleta (variables locales)
2. Conectamos unas tripletas con otras para formar un patrón
3. Aplicamos filtros (DataFrames) sobre el resultado

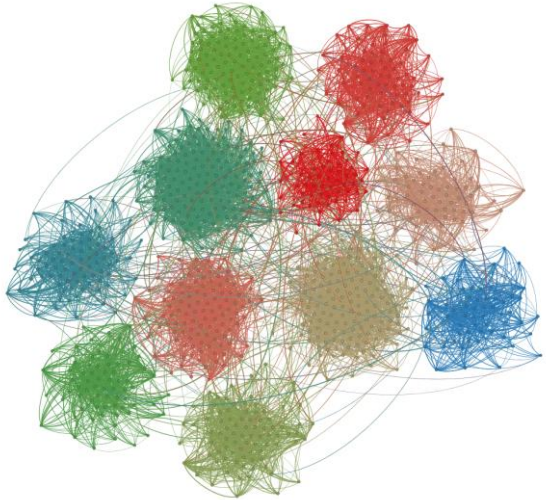
```
paths = g.bfs("name = 'Esther'", "age < 32")
paths.show()

g.bfs("name = 'Esther'", "age < 32",
  edgeFilter="relationship != 'friend'", maxPathLength=3)
```

Detección de comunidades

- En una red compleja es frecuente encontrar una estructura de comunidades
  - regiones fuertemente conectadas separadas entre sí por enlaces menos densos
- Para detectarlas se puede recurrir a distintos algoritmos de clustering, inferencia estadística, cliques, etc
- GraphFrames contiene un algoritmo de detección de comunidades (LPA – Label Propagation Algorithm)

Búsqueda de comunidades

- Algoritmo de propagación de etiquetas (LPA)



- Cómo encontrar grupos de nodos con un cierto agrupamiento en un grafo (clustering)

Problemas del algoritmo LPA

- No está garantizada su convergencia
- En función de los parámetros iniciales puede llegar a soluciones triviales que no aportan información => Ej: una sola comunidad

PageRank

PageRank: Orígenes (I)

Journal Summary List

Journals from: subject categories CELL BIOLOGY [VIEW CATEGORY SUMMARY LIST]

Sorted by: Impact Factor [SORT AGAIN]

Journals 1 - 20 (of 157) [1 | 2 | 3 | 4 | 5 | 6 | 7 | 8]

[MARK ALL] [UPDATE MARKED LIST]

Ranking is based on your journal and sort selections.

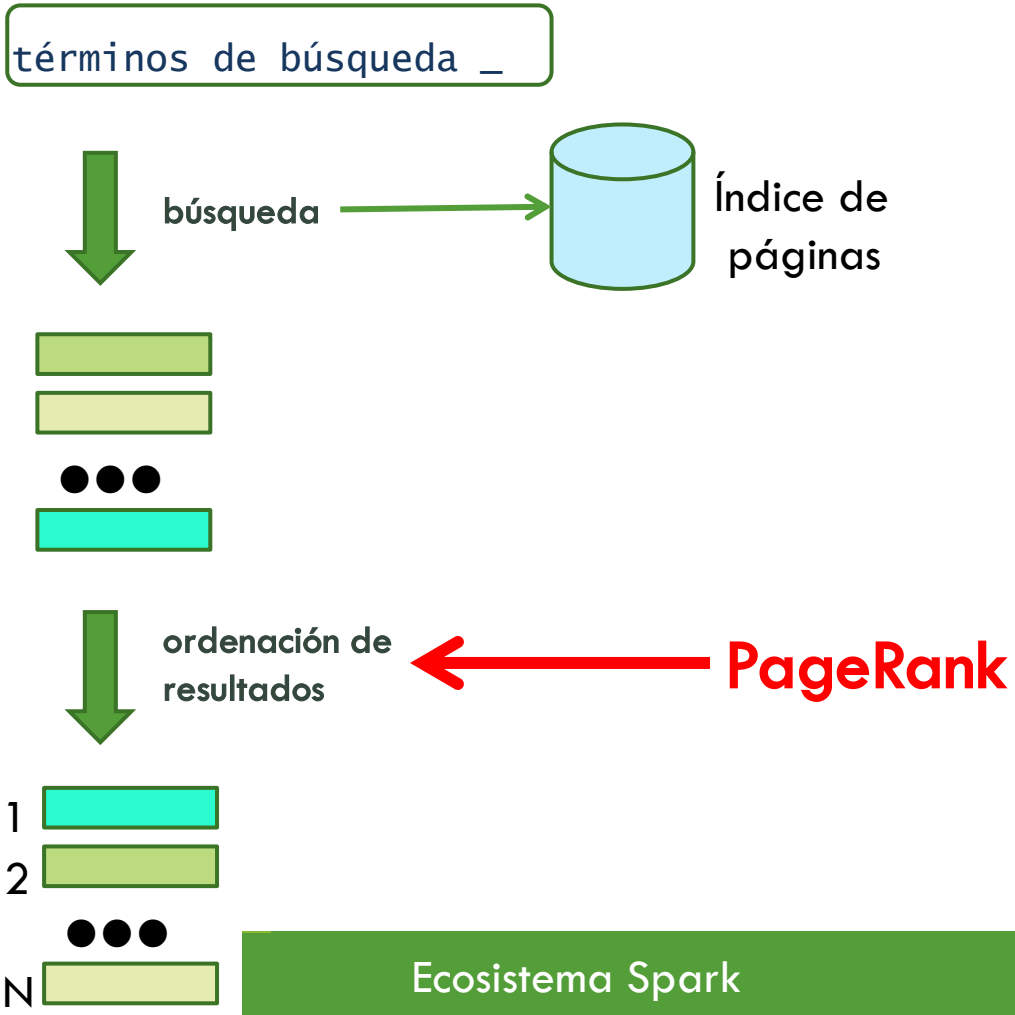
Mark	Rank	Abbreviated Journal Title (linked to journal information)	ISSN	JCR Data						Eigenfactor
				Total Cites	Impact Factor	5-Year Impact Factor	Immediacy Index	Articles	Cited Half-life	Eigenfactor™ Score
☐	1	NAT REV MOL CELL BIO	1471-0072	19628	35.423	34.221	7.238	84	4.0	0.17836
☐	2	CELL	0092-8674	142064	31.253	30.149	6.126	348	8.8	0.67169
☐	3	NAT MED	1078-8956	48632	27.553	28.965	5.546	141	6.1	0.22687
☐	4	ANNU REV CELL DEV BI	1081-0706	8063	22.731	26.058	1.167	24	7.1	0.04961
☐	5	NAT CELL BIOL	1465-7392	22676	17.774	17.637	5.145	145	5.2	0.16986
☐	6	CELL STEM CELL	1934-5909	1232	16.826	16.826	4.760	96	1.2	0.01254
☐	7	CELL METAB	1550-4131	4463	16.107	17.974	3.653	98	2.5	0.04804

Medición de impacto de publicaciones académicas.

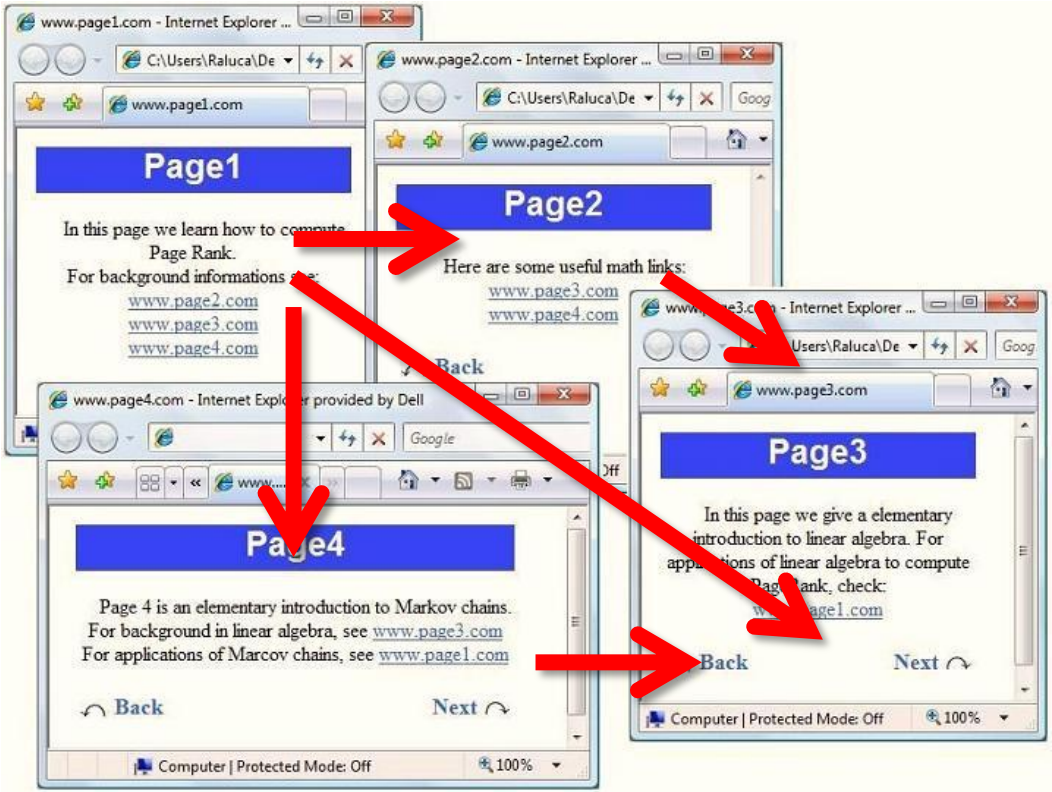
Pinski, G., & Narin, F. (1976). Citation influence for journal aggregates of scientific publications: Theory, with application to the literature of physics. *Information Processing & Management*, *12*(5), 297-312.

PageRank: Orígenes (II)

Principio básico del buscador



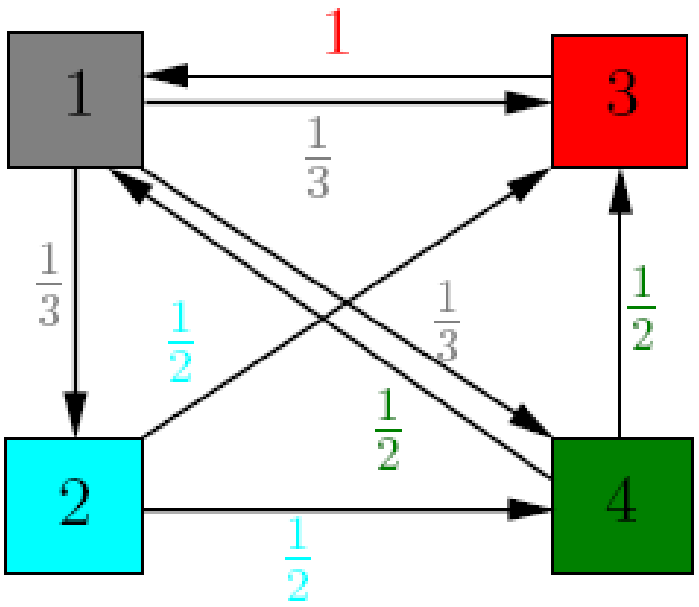
PageRank: Interpretación (I)



PageRank: Interpretación (II)

- PageRank de una página: Probabilidad de que un surfista aleatorio que abre una web al azar y se pone a seguir enlaces, visite dicha página

PageRank: Interpretación (III)



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

PageRank: Visión como cadena de Markov

C	R	S
1	0	0

X

	C	R	S
C	0.1	0.5	0.4
R	0.3	0.6	0.1
S	0.4	0.1	0.5

=

C	R	S
1x0.1 + 0x0.3 + 0x0.4	1x0.5 + 0x0.6 + 0x0.1	1x0.4 + 0x0.1 + 0x0.5
0.1	0.5	0.4

- Si se cumple que la cadena de Markov es:
  - Irreducible: Si en la matriz se puede llegar de cualquier nodo a cualquier otro nodo
  - Aperiódica: No existen ciclos en el grafo

Entonces el proceso converge y la solución es única (!)

Cálculo de PageRank

PageRank de la página p_i

$$PR(p_i) = \frac{1-d}{N} + d \sum_{p_j \in M(p_i)} \frac{PR(p_j)}{L(p_j)}$$

Factor de suavizado: teleportado a una página arbitraria

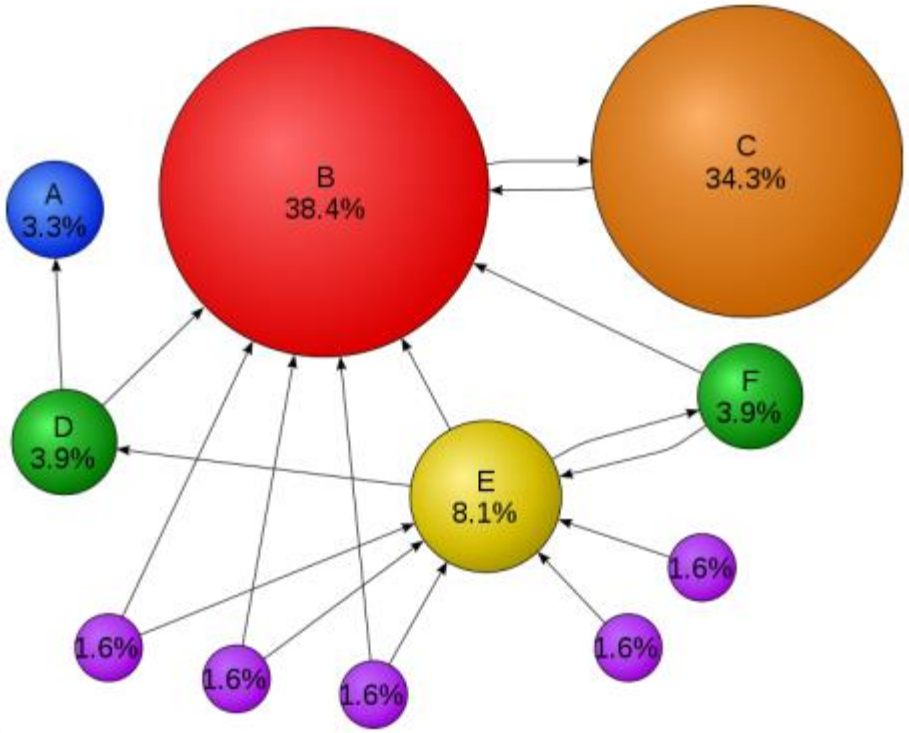
Suma sobre todas las páginas p_j que enlazan a p_i

Porcentaje de PageRank de p_j "entregado" a p_i

$$\mathbf{R} = \begin{bmatrix} PR(p_1) \\ PR(p_2) \\ \vdots \\ PR(p_N) \end{bmatrix}$$

$$\mathbf{R} = \begin{bmatrix} (1-d)/N \\ (1-d)/N \\ \vdots \\ (1-d)/N \end{bmatrix} + d \overbrace{\begin{bmatrix} \ell(p_1,p_1) & \ell(p_1,p_2) & \cdots & \ell(p_1,p_N) \\ \ell(p_2,p_1) & \ddots & & \vdots \\ \vdots & & \ell(p_i,p_j) & \\ \ell(p_N,p_1) & \cdots & & \ell(p_N,p_N) \end{bmatrix}}^{\text{Matriz de adyacencia normalizada}} \mathbf{R}$$

PageRank: Un ejemplo



PageRank en GraphFrames

- Dos posibles modos de ejecución:
  - Número fijo de iteraciones
  - Ejecutar hasta convergencia (con factor de tolerancia)

```
results = g.pageRank(resetProbability=0.15, maxIter=10)
results = g.pageRank(resetProbability=0.15, tol=0.01)

results.vertices.select("id", "pagerank").show()
results.edges.select("src", "dst", "weight").show()
```

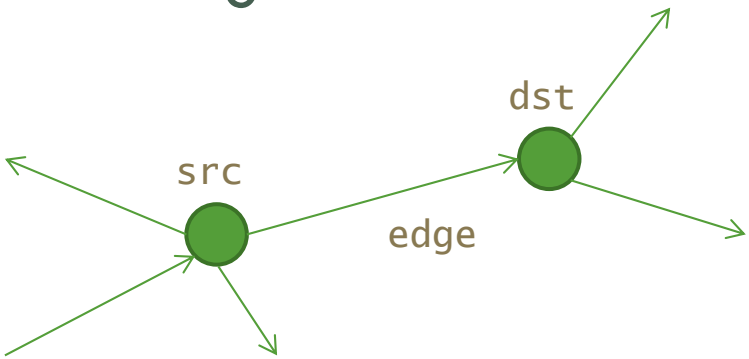
Agregación de mensajes

Desarrollo de algoritmos adicionales sobre GraphFrames

- Se pueden implementar mediante dos primitivas básicas, de forma iterativa
  - `aggregateMessages`: *send messages between vertices, and aggregate messages for each vertex.*
  - `joins`: *join message aggregates with the original graph.*
- Ambas están implementadas sobre operaciones de DataFrames

- `Pregel` es también un algoritmo iterativo de paso de mensajes, con un nivel algo mayor de abstracción

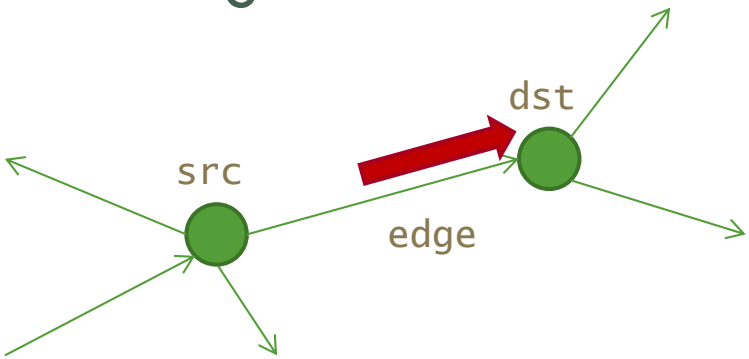
AggregateMessages



ejecutado sobre tripletas
(src, edge, dst)

```
messages = graph.aggregateMessages(aggCol, sendToSrc, sendToDst)
```

AggregateMessages

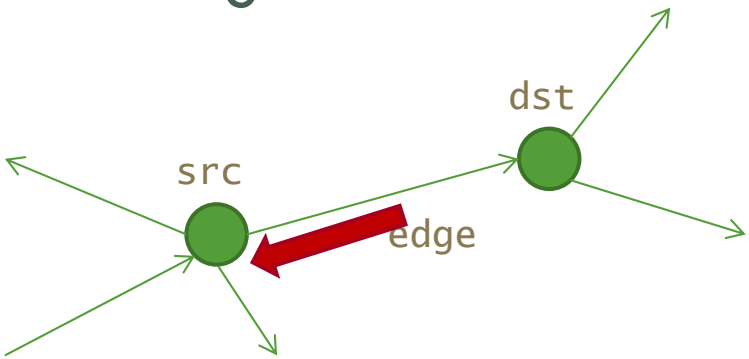


ejecutado sobre tripletas
`(src, edge, dst)`

Envía mensajes al nodo **dst**

```
messages = graph.aggregateMessages(aggCol, sendToSrc, sendToDst)
```

AggregateMessages

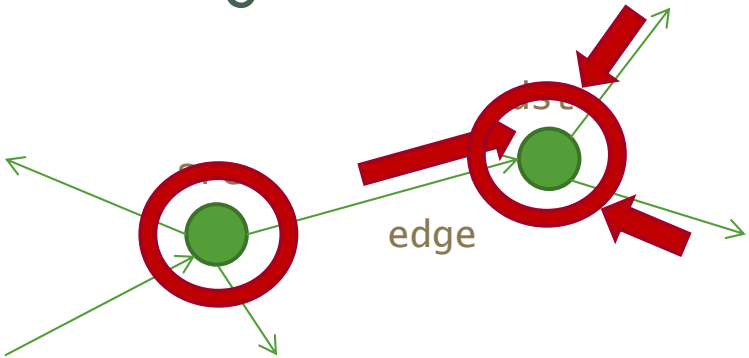


ejecutado sobre tripletas
(src, edge, dst)

Envía mensajes al nodo **src**

```
messages = graph.aggregateMessages(aggCol, sendToSrc, sendToDst)
```

AggregateMessages

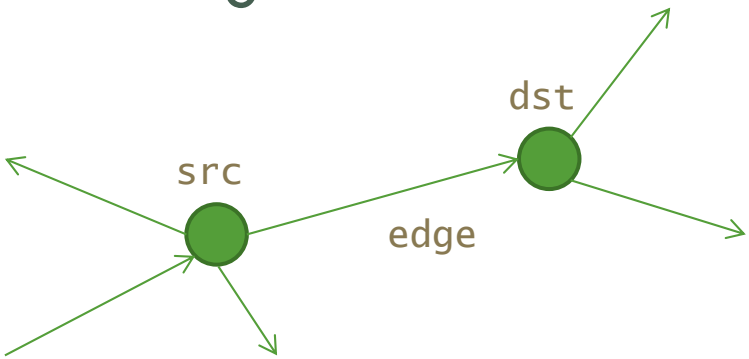


ejecutado sobre tripletas
(src, edge, dst)

Agrega los mensajes recibidos en cada nodo

```
messages = graph.aggregateMessages(aggCol, sendToSrc, sendToDst)
```

AggregateMessages



ejecutado sobre tripletas
(src, edge, dst)

```
messages = graph.aggregateMessages(aggCol, sendToSrc, sendToDst)
```

reduce | map | map

Algoritmo para AggregateMessages

Iteración de:

1. Recorre todo el grafo y para cada tripleta `(src, edge, dst)` envía mensajes:
   - `SendToDst`: mensajes enviados al nodo destino, usando datos de la tripleta
   - `SendToSrc`: mensajes enviados al nodo origen, usando datos de la tripleta
2. Recorre todo el grafo, y para cada nodo agrega los mensajes recibidos: `aggCol`
3. Guarda el resultado de agregar en el grafo (*join* con el df de vértices)
4. Comprueba condición de parada
5. Rehace el grafo y repite

Recomendaciones de proceso

Recomendaciones de proceso

- Recordar la exigencia de GraphFrames: **id, src, dst**
  - Es recomendable además usar <u>enteros</u> como índices
- Las operaciones complejas con grafos exigen mucho proceso en el cluster (iteraciones, *shuffles*)
  - Se puede controlar el grado de particionado en shuffles con `spark.sql.shuffle.partitions`
- Para algunas operaciones es necesario *checkpointing*
  - *checkpointing* más frecuente ➔ más seguro & más lento
- El rendimiento puede mejorar si se reparticionan los DataFrames
  - Particionar los enlaces por **src**, los nodos por **id**