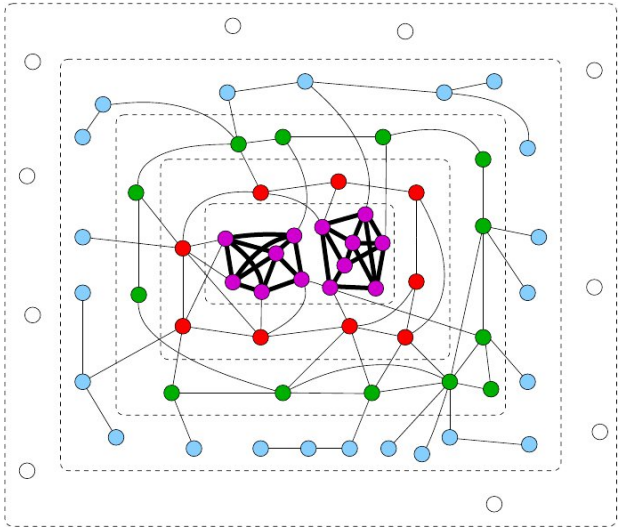
Graph Mining - I



F. Malliaros, M. Vazirgiannis

LIX @ Ecole Polytechnique

Outline

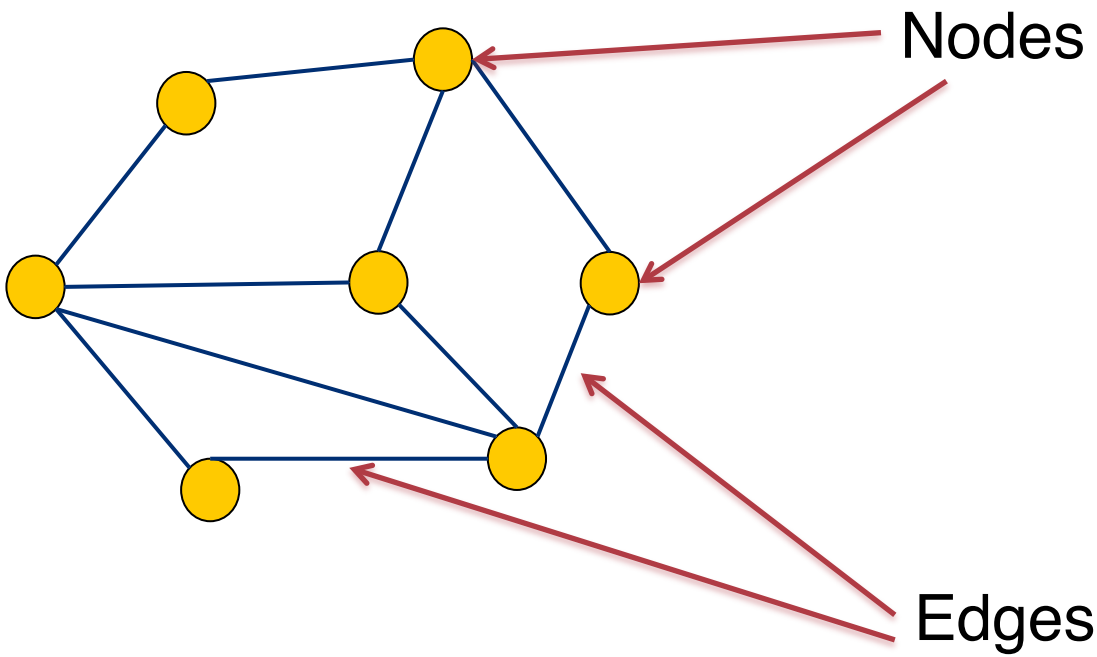
1. **Introduction & Motivation**
2. Graph Generators
3. Supervised Learning for graphs
    1. Graph Kernels – graph embeddings
    2. Graph classification
4. Unsupervised learning
    1. Community detection

Learning for Graphs Data

- **Data mining and machine learning have rich history and methods for analyzing**
  - Tabular data
  - Textual data
  - Time series
  - Market baskets
- **What about relations and dependencies?**

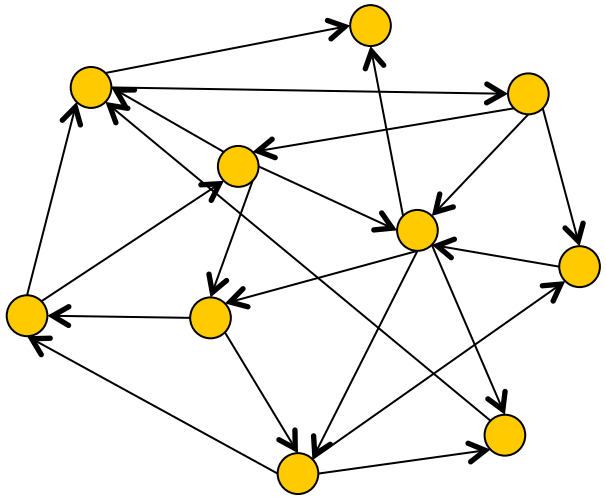
Graphs and Networks

- Graphs allow for modeling dependencies

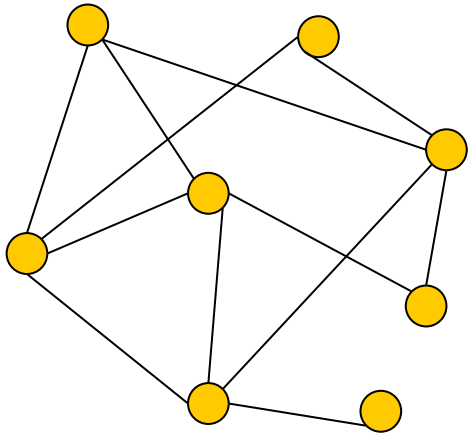


Basic Graph Definitions

- A graph **G=(V, E)** consists of a set of **nodes V, |V|= n** and a set of **edges E, |E| = m**
- Graphs can be **undirected** or **directed**



Directed

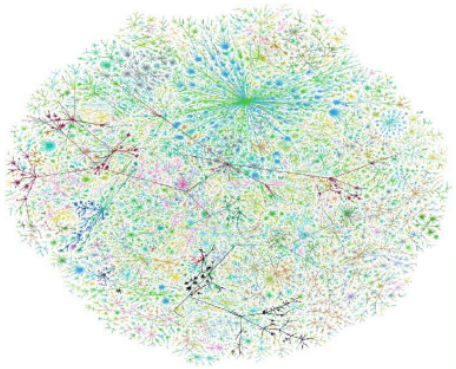


Undirected

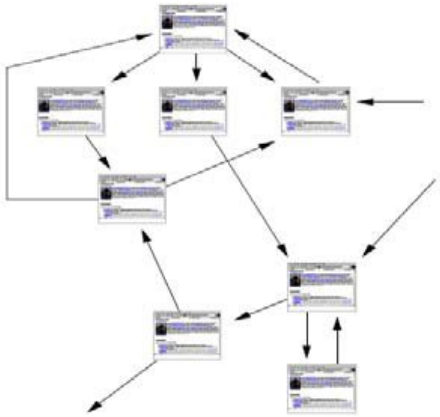
In-degree: $d_{in}(i) = ||\ j\ |\ (j,i)$ **is edge** $||$
Out-deg: $d_{out}(i) = ||\ j\ |\ (i,j)$ **is edge** $||$

Degree: $d(i) = d_{in}(i) = d_{out}(i)$

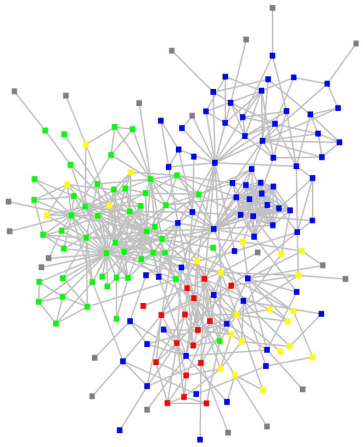
Networks are Everywhere



Internet

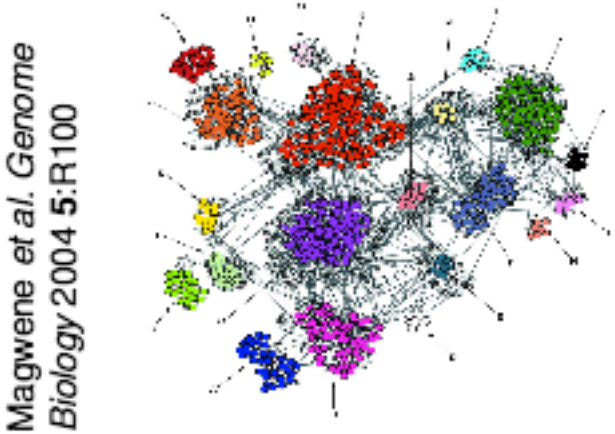


World Wide Web

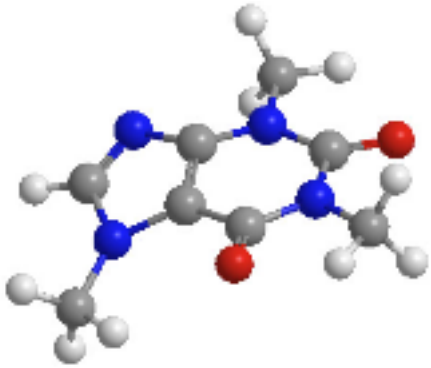


Email network

Social network

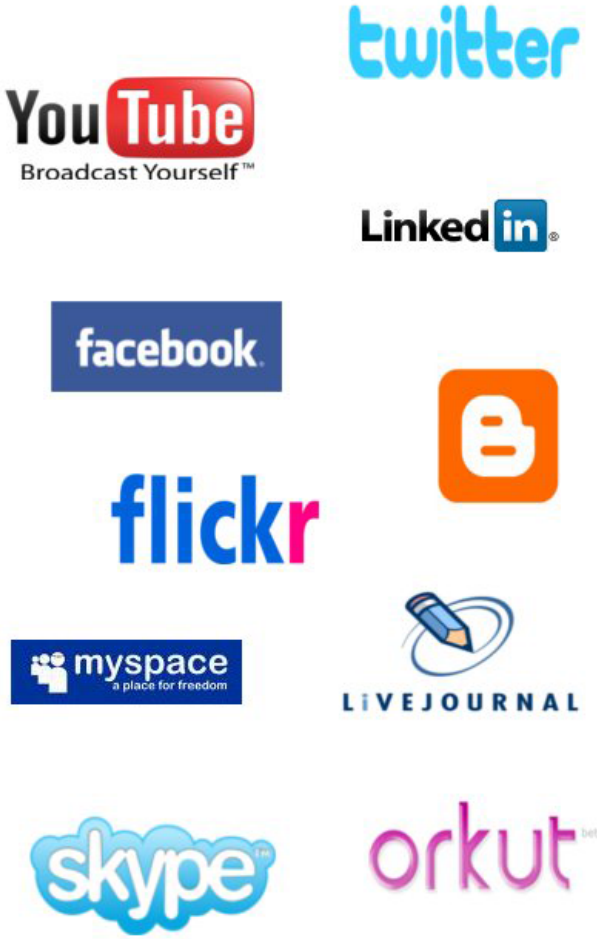


Co-expression network



Chemical network

Social Networking Data



- Online social networks and social media
- Easily accessible network data at **large scale**
- Opportunity to scale up observations
- Large amounts of data raise new questions

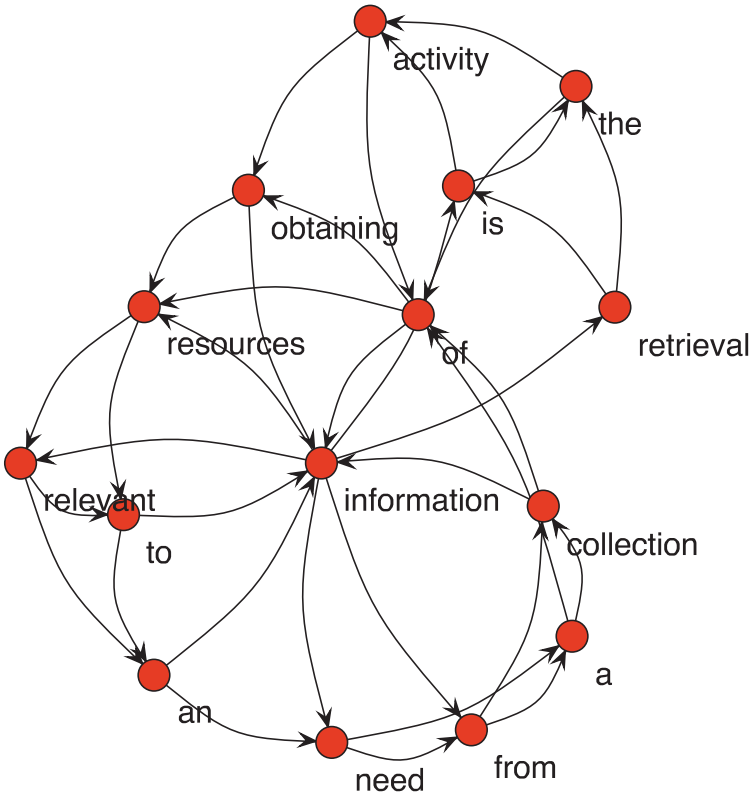
7

Even representing text - Graph-of-word

information retrieval is the activity of obtaining

information resources relevant to an information need

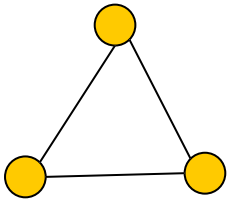
from a collection of information resources



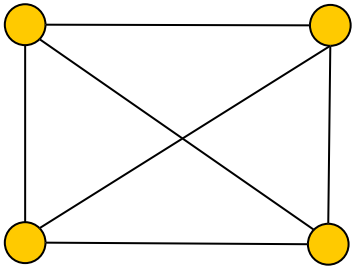
"Graph of word approach for ad-hoc information retrieval", F. Rousseau, M. Vazirgiannis, Best paper mention award ACM CIKM 2013

Complete Graph

- **Definition:** A graph **G=(V, E)** is called complete K_n if every pair of nodes is connected by an edge



Complete graph with 3 nodes: triangle



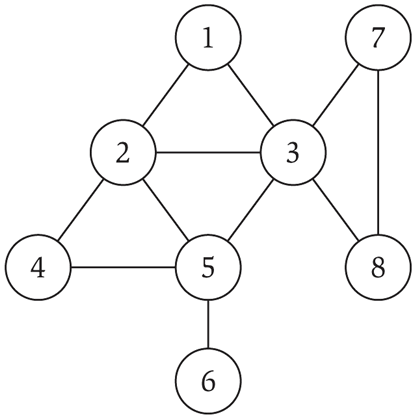
Complete graph with 4 nodes

- What is the number of edges of a complete graph with n nodes?

- Note that, the notion of complete graphs is of particular importance for the problem of community detection
  - **Communities correspond to well-connected subgraphs**

Graph Representation: Adjacency Matrix

- A graph can be represented by the adjacency matrix **W**
  - Matrix of size **n x n**, where **n** is the number of nodes
  - $W_{ij} > 0$, if **i** and **j** are connected
  - $W_{ij} = 0$, if **i** and **j** are not connected
  - In case of unweighted graphs, $W_{ij} = 1$, if **(i, j)** is an edge of the graph
  - Space proportional to n^2



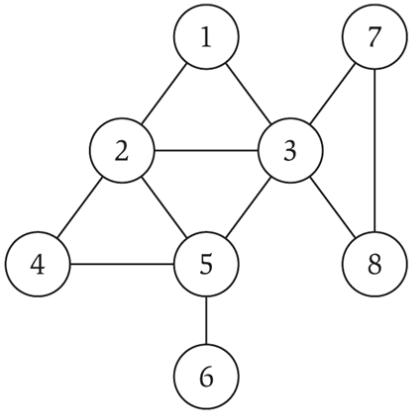
Undirected graph

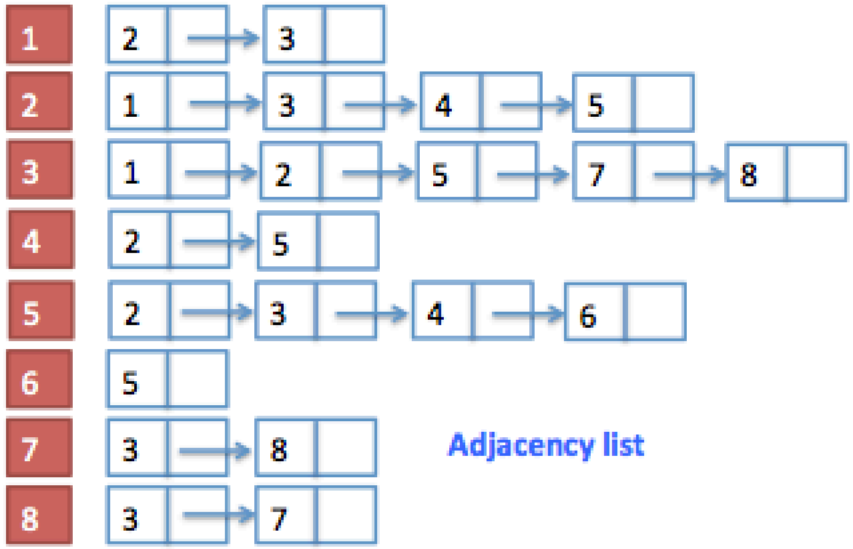
0	1	1	0	0	0	0	0
1	0	1	1	1	0	0	0
1	1	0	0	1	0	1	1
0	1	0	0	1	0	0	0
0	1	1	1	0	1	0	0
0	0	0	0	1	0	0	0
0	0	1	0	0	0	0	1
0	0	1	0	0	0	1	0

Adjacency matrix

Graph Representation: Adjacency Lists

- Adjacency lists
  - Representation of a graph with **n** nodes using an array of **n** lists of nodes
  - List **i** contains node **j** if there is an edge **(i, j)**
  - A weighted graph can be represented with a list of node/weight pairs
  - Space proportional to **Θ(m+n)**
  - Checking if **(i, j)** is an edge takes **$O(d_i)$** time

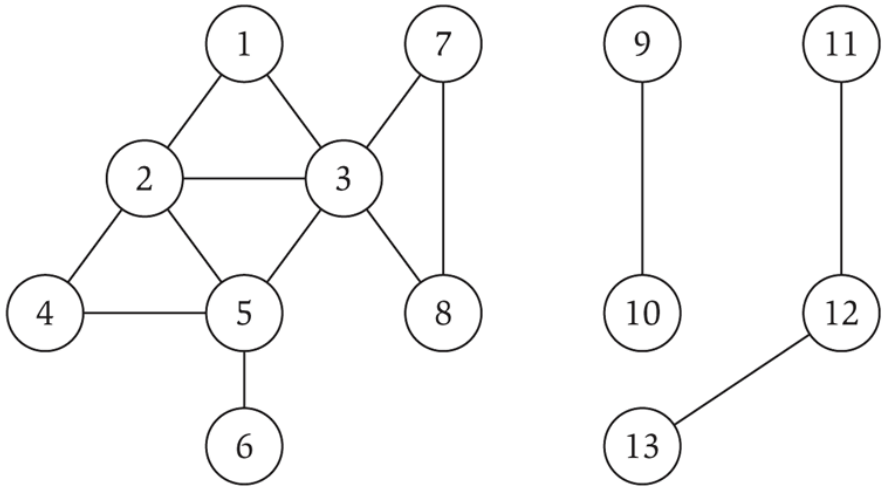




Paths and Connectivity in Graphs

- **Definition:** A path in an undirected graph **G=(V,E)** is a sequence of nodes $v_1, v_2, \ldots, v_k$ with the property that each consecutive pair v_{i-1}, v_i is joined by an edge in **E**

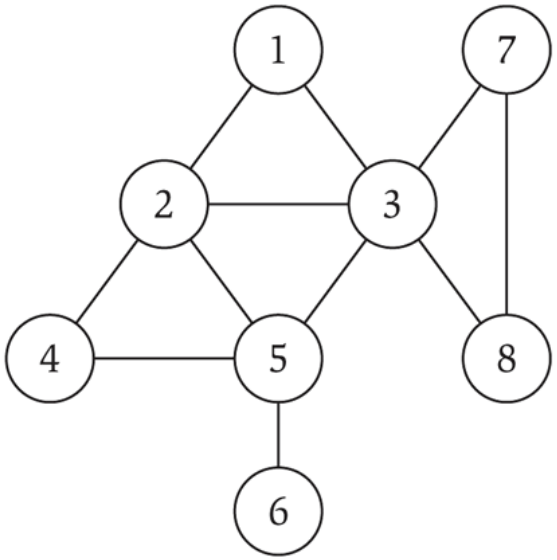
- **Definition:** An undirected graph is connected if for every pair of nodes **u** and **v**, there is a path between **u** and **v**



12

Cycles in Graphs

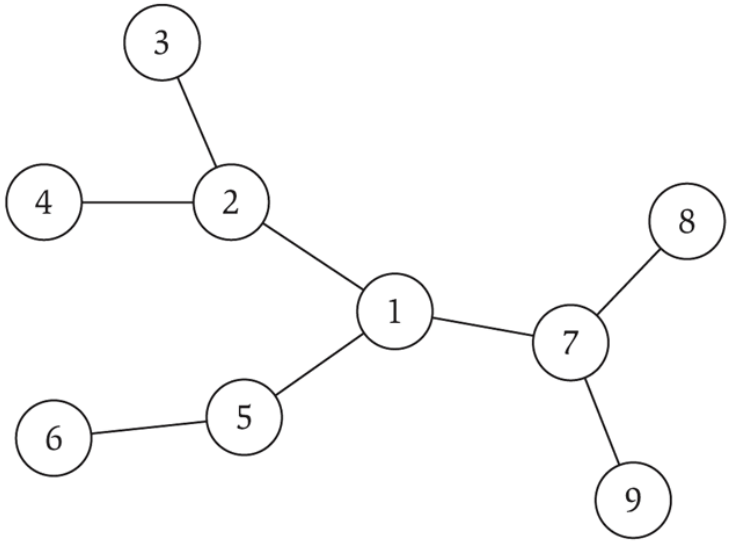
- **Definition:** A cycle is a path $v_1, v_2, \ldots, v_k$ in which $v_1 = v_k$, $k > 2$ and the first $k-1$ nodes are all distinct



Cycle C = 1 – 2 – 4 – 5 – 3 – 1

Trees

- **Definition:** An undirected graph is a tree if it is connected and does not contain a cycle
- **Theorem:** Let **G** be an undirected graph with **n** nodes. Then, any two of the following statements imply the third:
  - **G** is connected
  - **G** does not contain a cycle
  - **G** has **n-1** edges



14

Graph Traversal

- **Graph traversal** is the problem of visiting all the nodes in the graph in a particular manner

- Graph traversal algorithms
  - **Breadth-first search (BFS)**
  - **Depth-first search (DFS)**

15

Breadth-First Search (BFS)

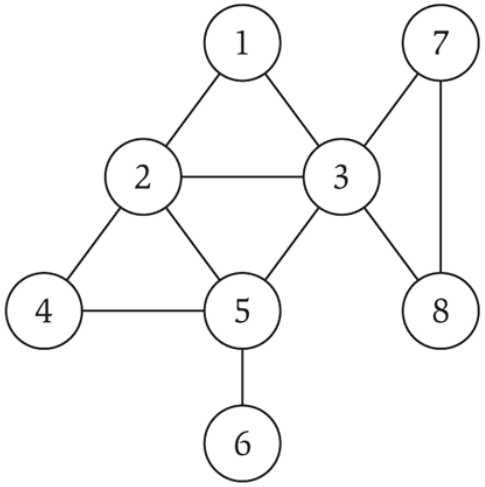
- Strategy for searching in graphs when search is limited to two operations:
  - Visit and inspect a node of a graph
  - Gain access to visit the nodes that neighbor the currently visited node

```
1  procedure BFS(G, v):
2      create a queue Q
3      add v onto Q
4      mark v
5      while Q is not empty:
6          t ← Q.dequeue()
7          if t is what we are looking for:
8              return t
9          for all edges e in G.adjacentEdges(t) do
12             o ← G.adjacentVertex(t,e)
13             if o is not marked:
14                 mark o
15                 add o onto Q
16     return null
```

Input: Graph **G** and a node **v**

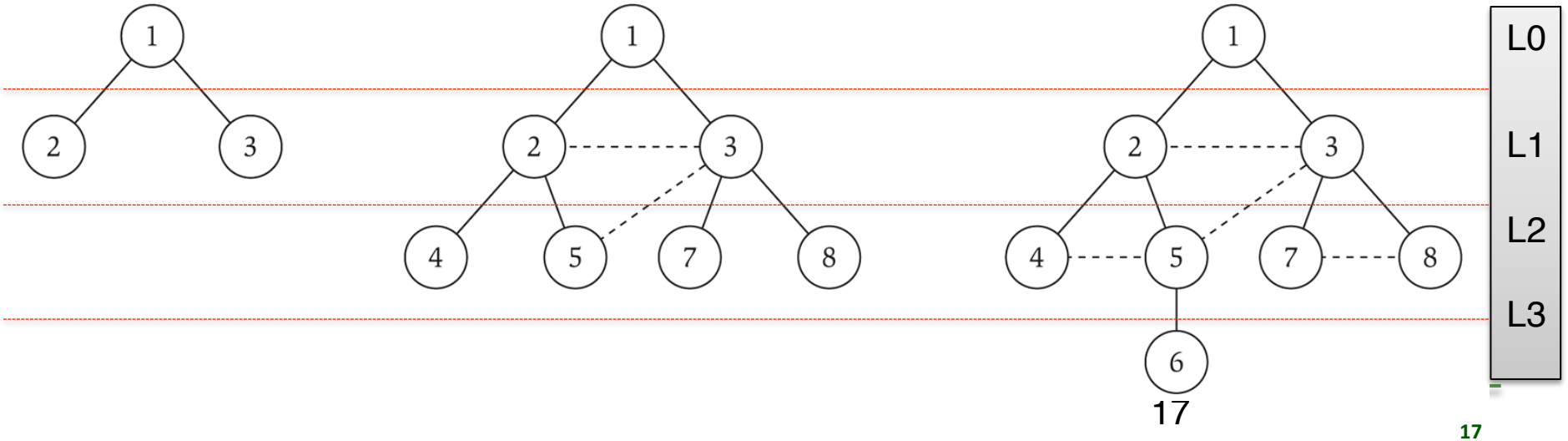
Output: An assignment (labeling) of the nodes of the graph into layers (or the node closest to **v** in **G** satisfying some conditions)

Breadth-First Search (BFS) - Example



Start BFS process from node 1

Property: Let T be a **BFS** tree of **G** and let **(u, v)** be an edge of **G**. Then, the level of **u** and **v** differ by at most 1



17

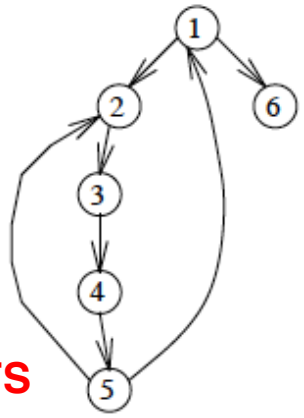
Depth-First Search (DFS)

- Strategy for traversing graphs:
  - Visits the child nodes before visiting the sibling nodes
  - Traverses the depth of any particular path before exploring the breadth

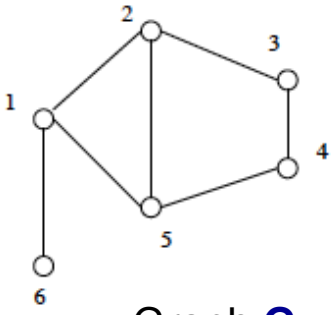
```
1  procedure DFS(G,v):
2      label v as explored
3      for all edges e in G.adjacentEdges(v)
do
4          if edge e is unexplored then
5              w ← G.adjacentVertex(v,e)
6              if vertex w is unexplored then
7                  label e as a discovery edge
8                  recursively call DFS(G,w)
9              else
10                 label e as a back edge
```

Input: Graph **G** and a node **v**

Output: A labeling of the edges of the graph as discovery and back edges



DFS

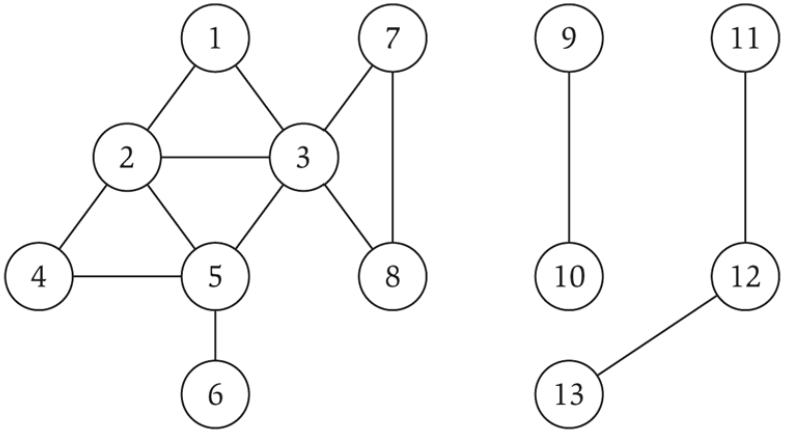


Graph **G**

18

Connected Components

- A **connected component** is a maximal connected subgraph of a graph **G** (there is a path between any pair of nodes)



Connected component containing node 1:
{1, 2, 3, 4, 5, 6, 7, 8}

Graph with 3 connected components

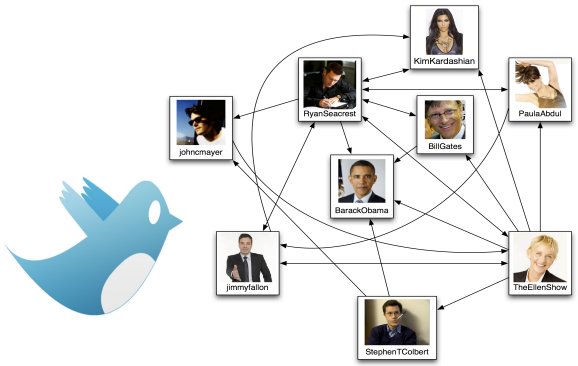
Question: How can we compute the connected components of a graph?

A: Apply BFS

19

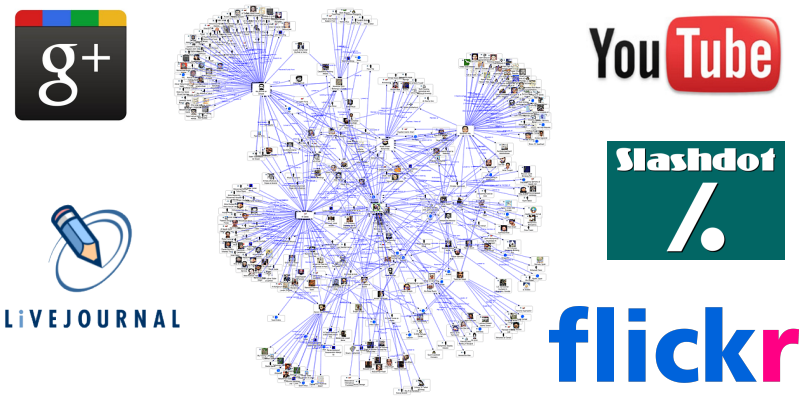
Connectivity in Directed Graphs (1/2)

- A plethora of network data from several applications is from their nature **directed**

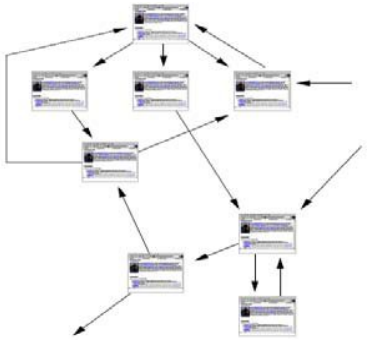


Twitter

[Image: http://sites.davidson.edu/mathmovement/]

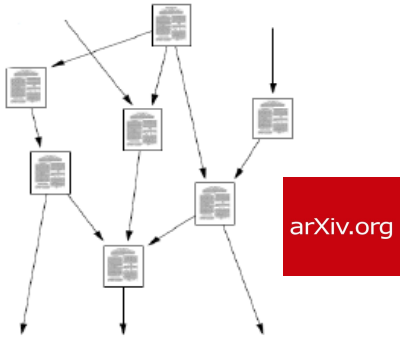


Online Social Networks



Web Graph

Wikipedia



Citation Graph

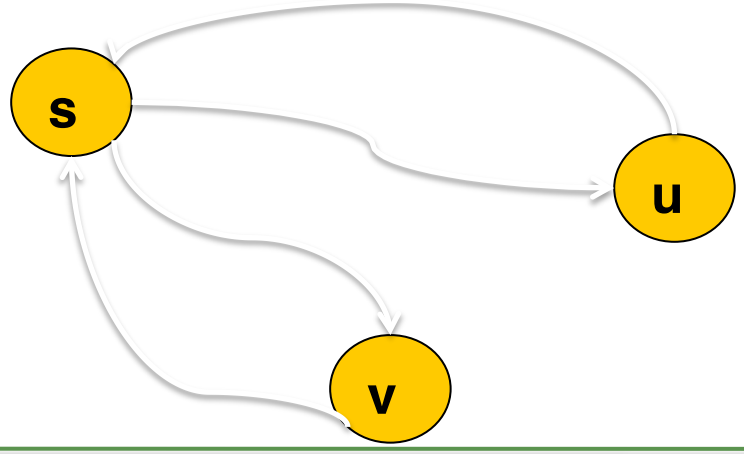
Connectivity in Directed Graphs (2/2)

- **Directed reachability:** Given a node **s** in a directed graph **G**, find all nodes reachable from **s**
  - **Web crawler:** start from a webpage s, find all webpages linked from s, either directly or indirectly
- The notion of connectivity in directed graphs is replaced by the
  - **Weak connectivity:** a graph is called weakly connected if it is possible to reach any node if we do not take into account the directionality of the edges
  - **Connectivity:** contains a directed path from **u** to **v** or a directed path from **v** to **u**, for every pair of nodes **u**, **v**
  - **Strong connectivity:** a graph is called strongly connected if it is possible to reach any node taking into account the directionality of the edges

Strong Connectivity

- **Lemma:** Let **s** be any node. Graph **G** is strongly connected iff every node is reachable from **s**, and **s** is reachable from every node
- **Proof:**
  - --> Follows from definition
  - <-- Path from **u** to **v**: concatenate **u --> s** path with **s --> v** path

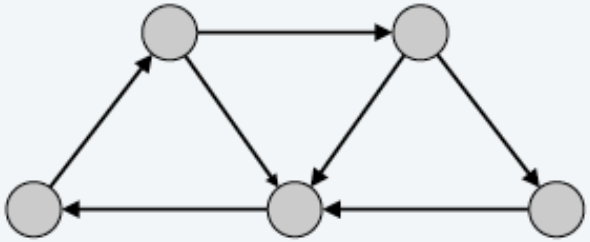
    Path from **v** to **u**: concatenate **v --> s** path with **s --> u** path



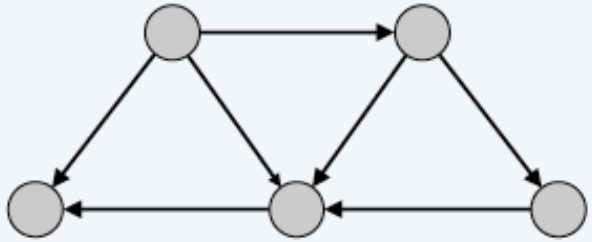
22

Strong Connectivity: Algorithm

- **Theorem:** We can determine if **G** is strongly connected in **O(m+n)** (i.e., linear) time using the BFS process
- **Proof:**
  - Pick any node **s**
  - Run **BFS** from **s** in **G**
  - Run **BFS** from **s** in $G_{reverse}$(reverse the orientation of edges)
  - Return true iff all nodes reached in both BFS executions



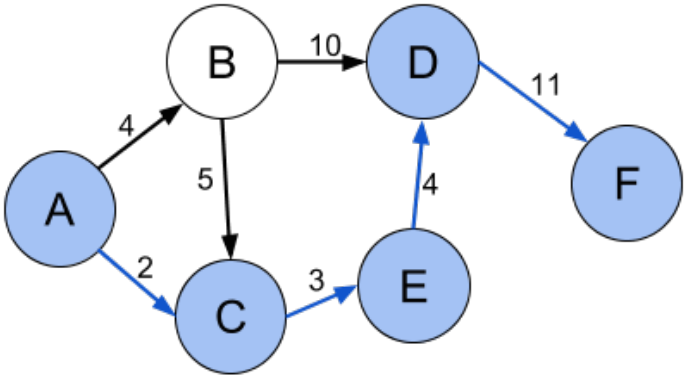
strongly connected



not strongly connected

Shortest Paths

- **Definition:** find a path between two nodes in a graph, in such a way that the sum of the weights of its constituent edges is minimized
  - Many applications (e.g., road networks)
  - **Single-source** shortest path problem
  - **Single-destination** shortest path problem
  - **All-pairs** shortest path problem

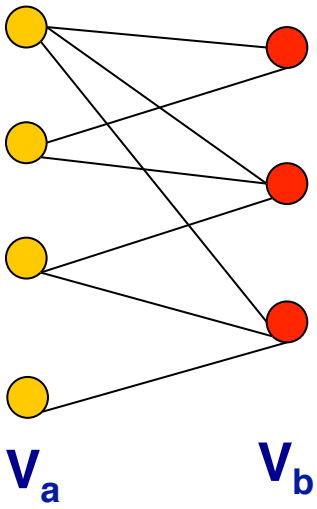


Shortest path (A, C, E, D, F) between vertices A and F in the weighted directed graph

Many algorithms:
- Dijkstra
- Bellman-Ford

Bipartite Graphs

- **Definition:** A graph **G=(V,E)** is called **bipartite** if the node set **V** can be partitioned into two disjoint sets V_a, V_b and every edge **(u,v)** connects a node of V_a to a node of V_b



- Strong modeling capabilities and many real-world applications
- E.g., **Collaborative filtering** in recommender systems
  - Model the customer-product space using a bipartite graph (who-purchased-what)
  - If a user A has purchased the same product with a user B, then it is more likely to purchase another product as B did, than of a person selected randomly

Learning for Graph Data: Some Challenges

- Highly **dynamic**: constantly changing in both **structure** and **size**
- **Multiple semantics/information** associated with the nodes and edges
  - Edges of the graphs
    - Unweighted or weighted
    - Undirected or directed (directed graphs)
    - Signed/trust networks (positive/negative interactions among individuals)

26

Outline

1. Introduction & Motivation
2. **Properties of real graphs**
3. Graph Generators
4. Supervised Learning for graphs
    1. Graph Kernels – graph embeddings
    2. Graph classification
5. Unsupervised learning
    1. Community detection

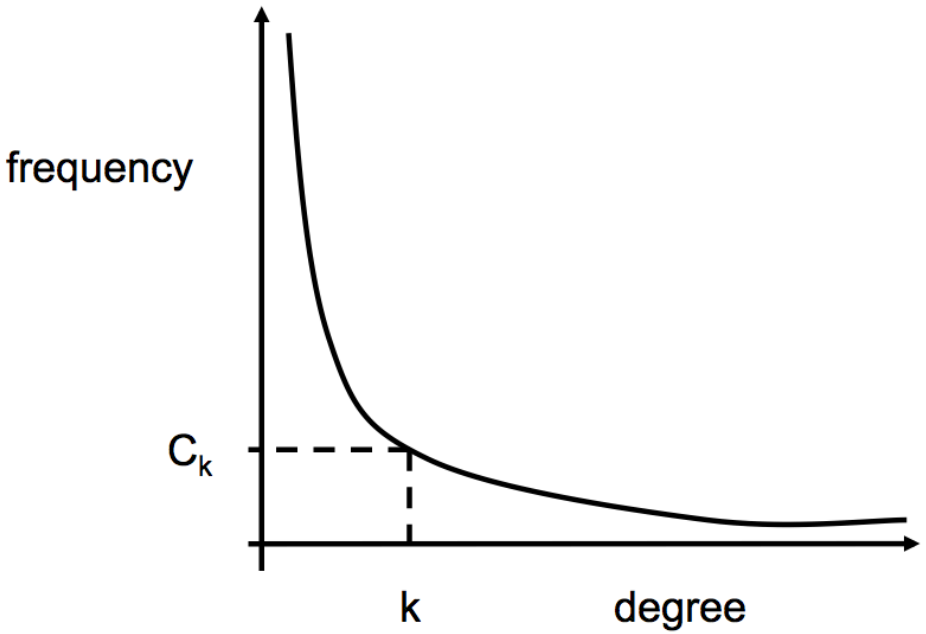
Properties of Real-World Graph

- Networks arising from **real-world** applications obey fascinating properties
- **Static networks**
  - Heavy-tailed degree distribution
  - Small diameter
  - Giant connected component (GCC)
  - Triangle Power Law
  - Community structure
  - …
- **Dynamic networks**
  - Densification
  - Small and shrinking diameter
  - …

28

Degree Distribution

- The **probability distribution** of the degrees over the network



- Let C_k = number of nodes with degree **k**
- **Problem:** find the probability distribution the **fits** best the **observed data**

29

Power-law Degree Distribution

- Let C_k = number of nodes with degree k

$$C_k = c\, k^{-\gamma}$$

with $\gamma > 1$ and c a constant

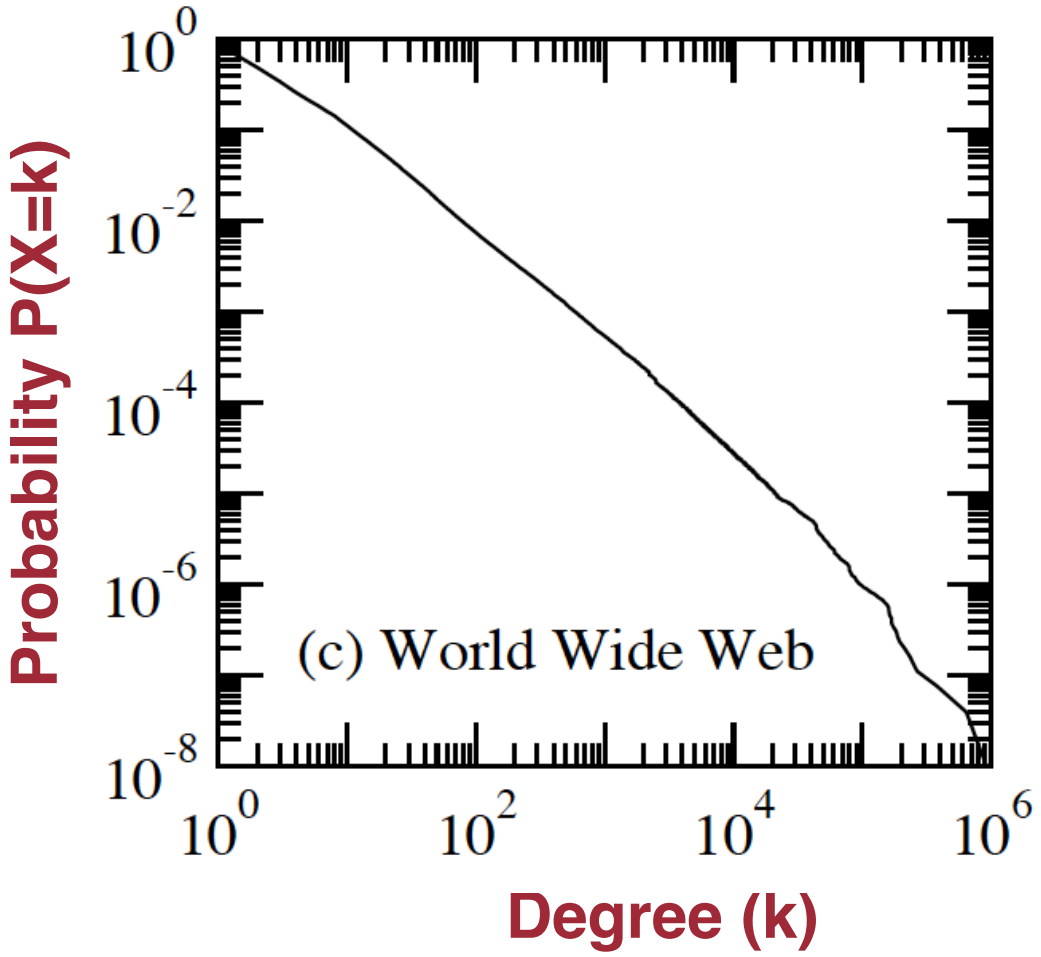
- How to recognize a power-law distribution?

$$\ln C_k = \ln c - \gamma \ln k$$

  - Plotting $\ln C_k$ versus $\ln k$ gives a straight line with slope $-\gamma \ln k$

30

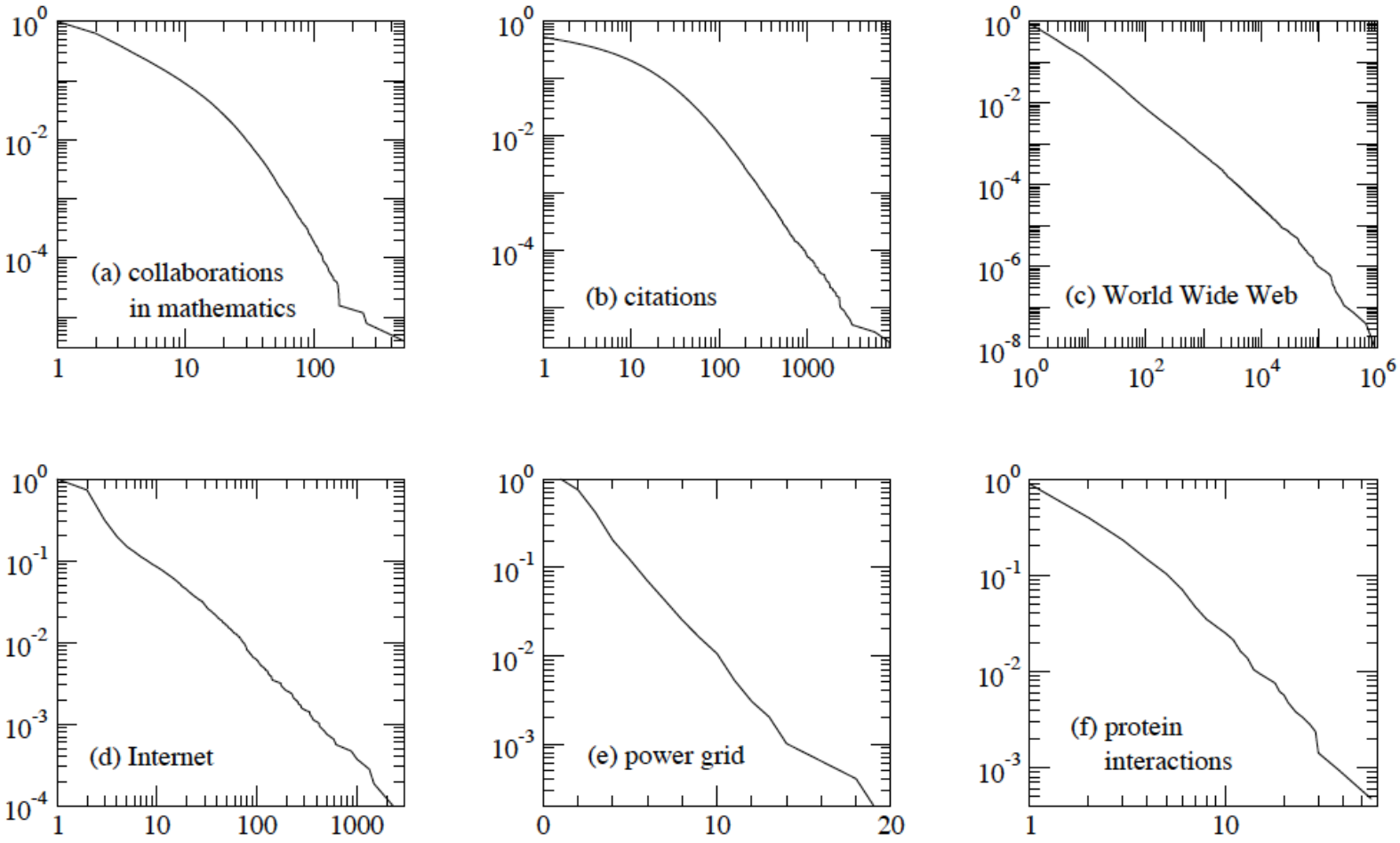
Power-law Degree Distribution in Real-Networks (1/2)



[Newman, 2003]

31

Power-law Degree Distribution in Real-Networks (2/2)



Cumulative degree distribution for six different networks [Newman 2003]

32

Power-law Degree Exponents

- **Power law degree exponent is typically $2 < \gamma < 3$**
  - Web graph [Broder et al., 2000]
    - $\gamma_{in} = 2.1$, $\gamma_{out} = 2.4$
  - Autonomous systems (Internet graph) [Faloutsos et al., 1999]
    - $\gamma = 2.4$
  - Actor collaborations [Barabasi and Albert, 2000]
    - $\gamma_{in} = 2.3$
  - Citation graphs [Redner, 1998]
    - $\gamma_{in} = 3$
  - MSN messenger graph [Leskovec et al., 2007]
    - $\gamma_{in} = 2$

[Lescovec, ICML, 2009]

Summary – Degrees in Real Networks

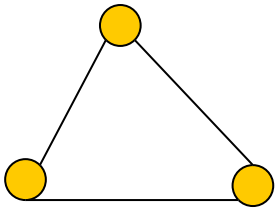
- The degree distribution is **heavily skewed**
  - Distribution is **heavy-tailed** (heavier tails compared to the exponential distribution)

$$\lim_{x \to \infty} \frac{Pr(X > x)}{e^{-\epsilon x}} = \infty$$

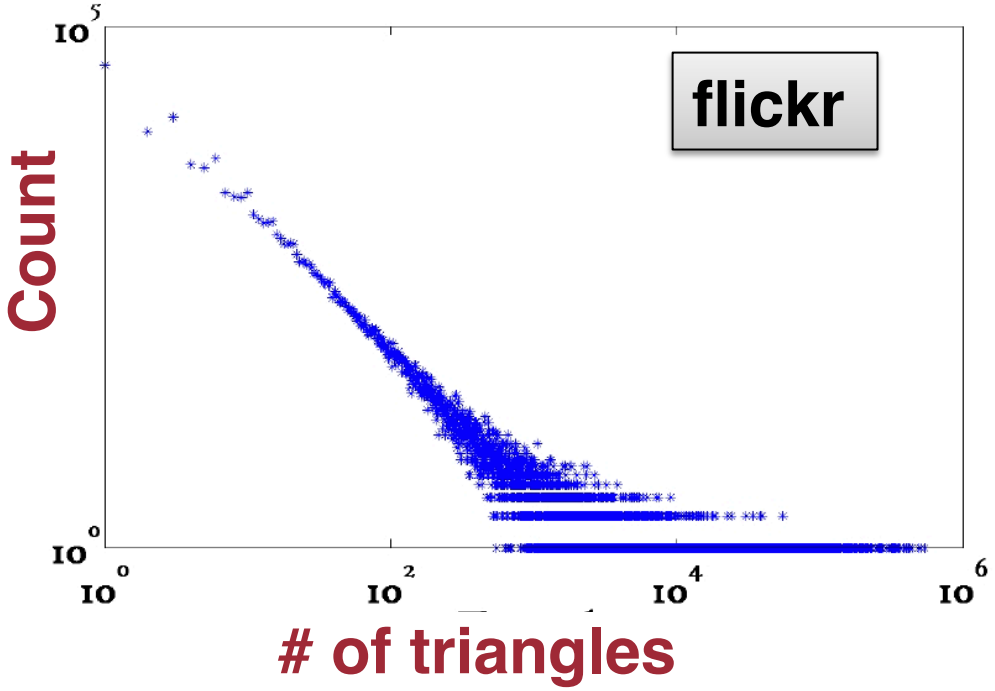
  - Various names and forms
    - Long tail, Zipf's law, Pareto distribution

34

Triangle Participation Distribution



Complete graph with 3 nodes: triangle



- Number of nodes that participate in **k** triangles vs. **k** in log-log scale
- **Heavy-tailed** distribution

[Tsourakakis, 2008]

35

Clustering Coefficient

- Captures the tendency of the nodes of a graph to cluster together

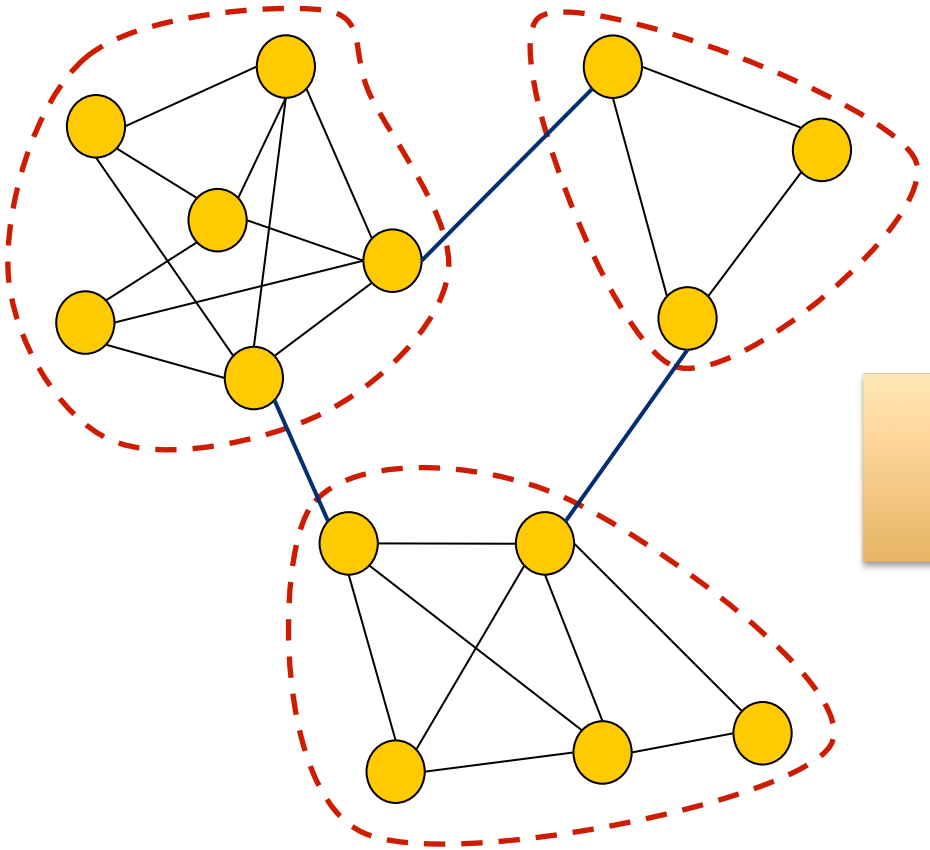
  T(G) = 3 x # of triangles in G / # of connected triplets

- Captures the transitivity of clustering
  - If **u** is connected to **v** and **v** is connected to **w** …
  - ... it is likely that **u** is also connected to **w**

- Real-world networks tend to have high clustering coefficient
  - Connections to the existence of clustering and community structure property

[Frieze, Gionis, Tsourakakis, 2013]

Community Structure

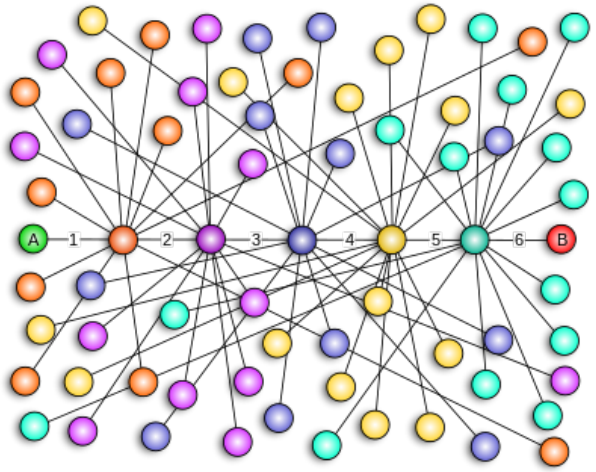


Example graph with three communities

- Will be covered later on in detail

Small-world Phenomenon (1/4)

- Six degrees of separation
  - Experiment done by sociologist Stanley Milgram (1960's)
  - Randomly selected people in Nebraska were asked to send letters to Boston, by contacting somebody with whom they had direct connection
  1. People either sent the letter directly to the recipient
  2. Or to somebody they believed had a high likelihood of knowing the target
- For those letters that reached their destination, the **average path length was 5.5 to 6**
  - Sort paths are abundant in the networks
  - **Decentralized routing:** people are capable of discovering which links to follow to reach faster the target



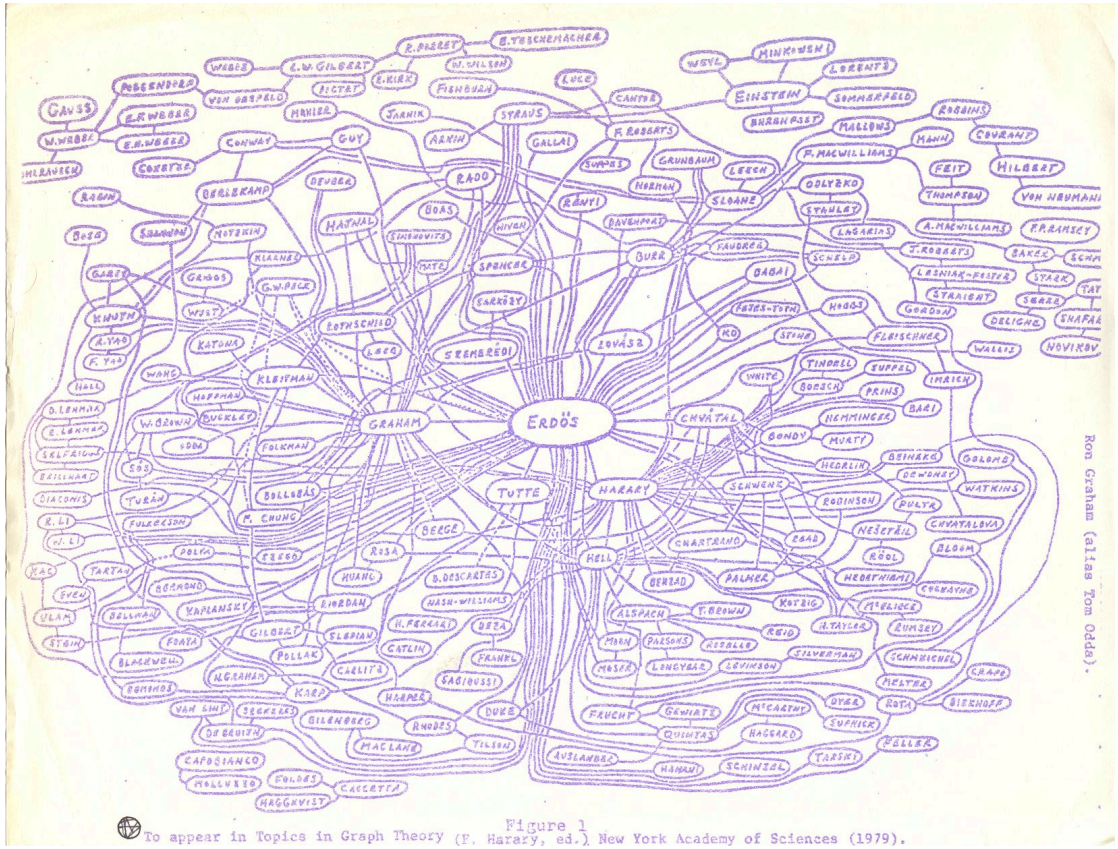
Source: Wikipedia

[McAuley, 2015]

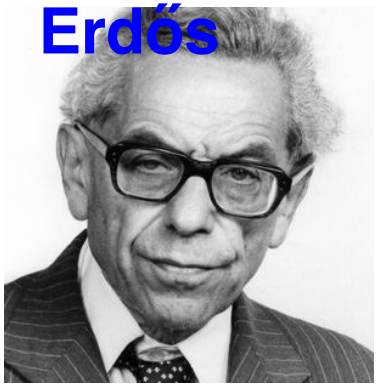
38

Small-world Phenomenon (3/4)

- The small-world phenomenon appears in various network settings



Paul Erdős



Source: UCSD

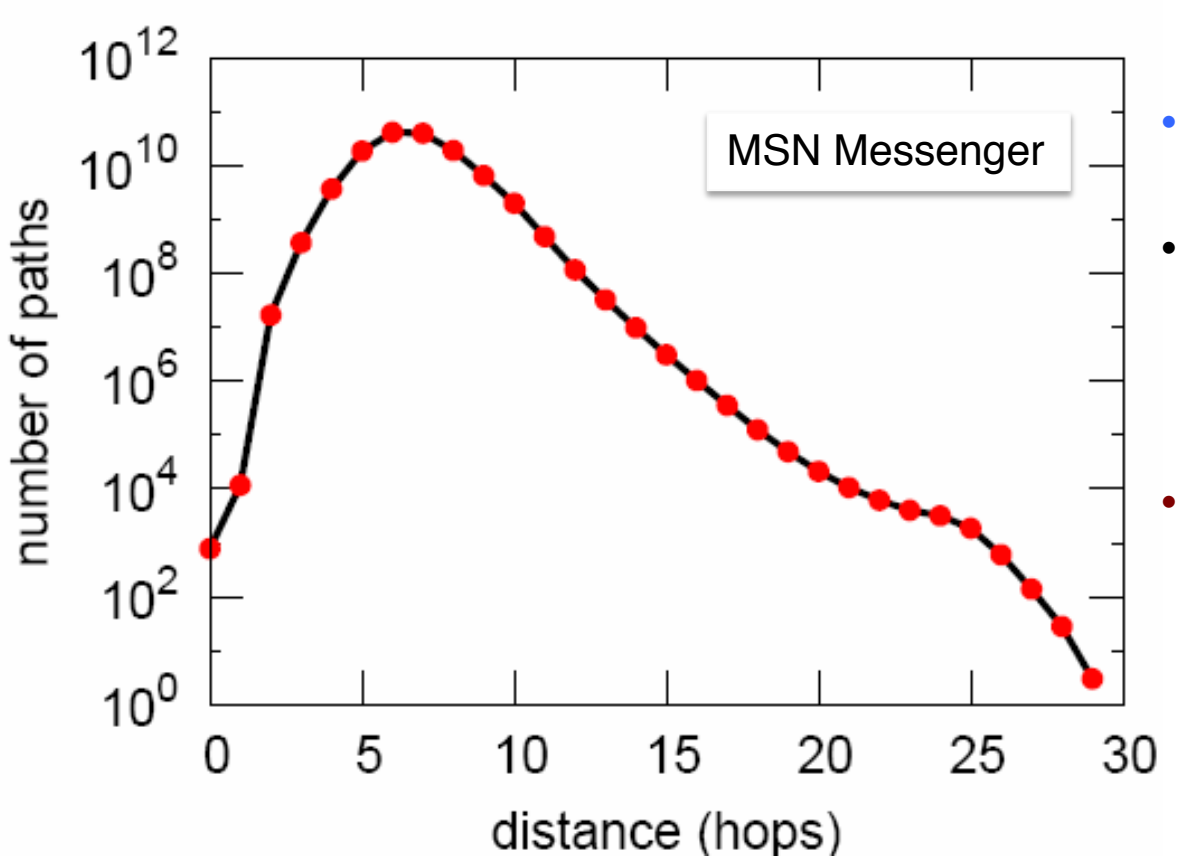
Source: physicsbuzz.physicscentral.com

Erdős number: # of hops needed to connect the author of a paper to Paul Erdős

39

Small-world Phenomenon (4/4)

- The small-world phenomenon appears in various network settings

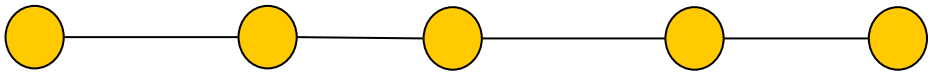


- **Average path length** is **6.6**
- 90% of the nodes are reachable in less than 8 steps
- **Facebook** network:
  - Average distance is **4.7**
  - **[Ugander et al., 2011]**

[Leskovec, 2009]

Small Diameter

- **Diameter** is the largest shortest path in the graph
  - Diameter is often sensitive to **chains** of nodes



- In practice, we use the **effective diameter**
  - Upper bound of the shortest path over 90% of the pairs of nodes

- As an effect of the small-world phenomenon, real networks have **small diameter**

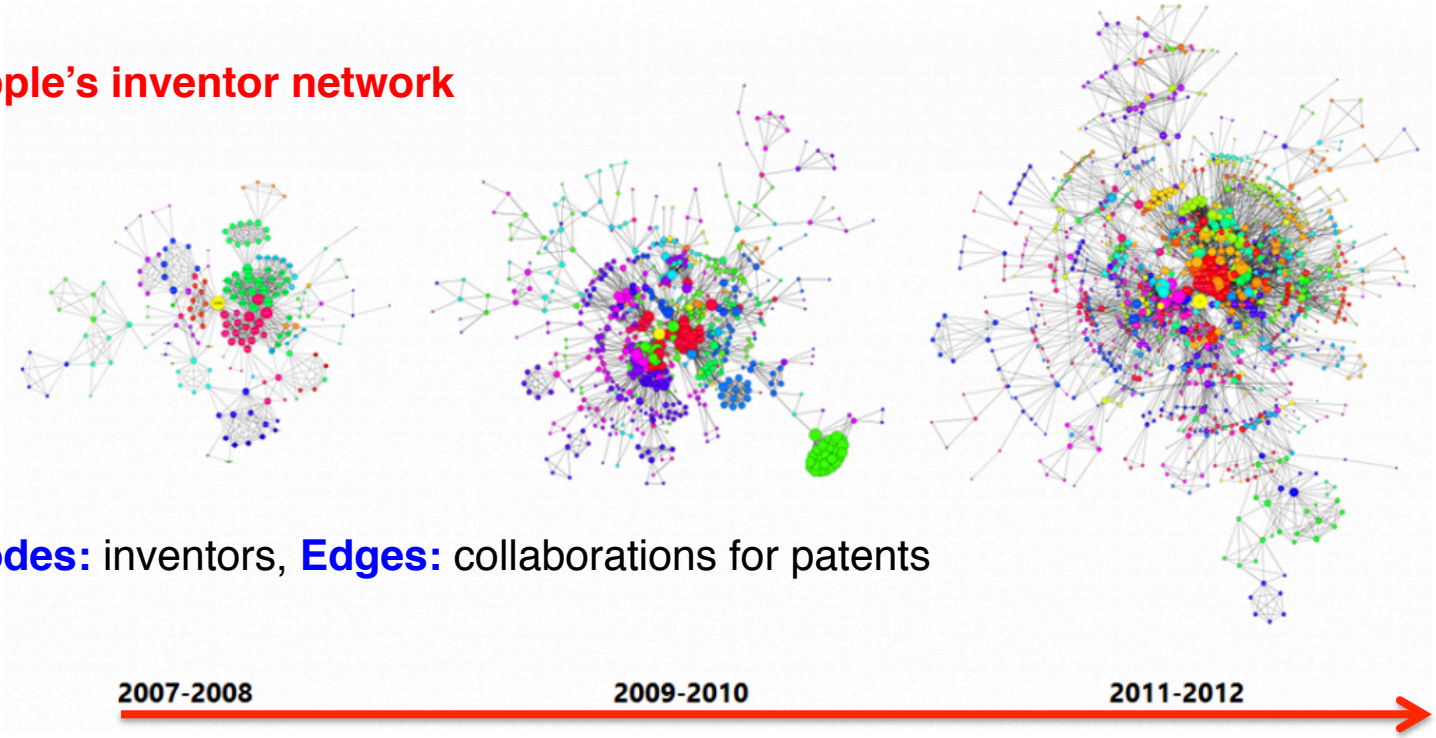
[Leskovec, 2009]

41

Network Evolution

- Real-world networks are not static, but they evolve over time
  - New nodes/edges are added and/or deleted
  - We are interested in making predictions about the structure of the network

Apple's inventor network



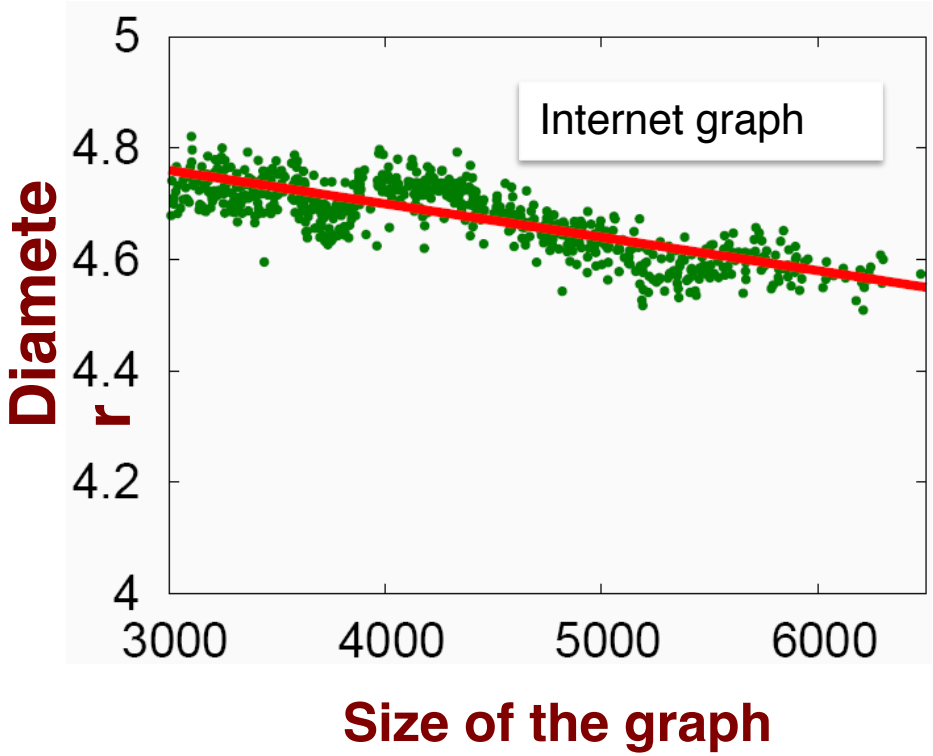
Nodes: inventors, **Edges:** collaborations for patents

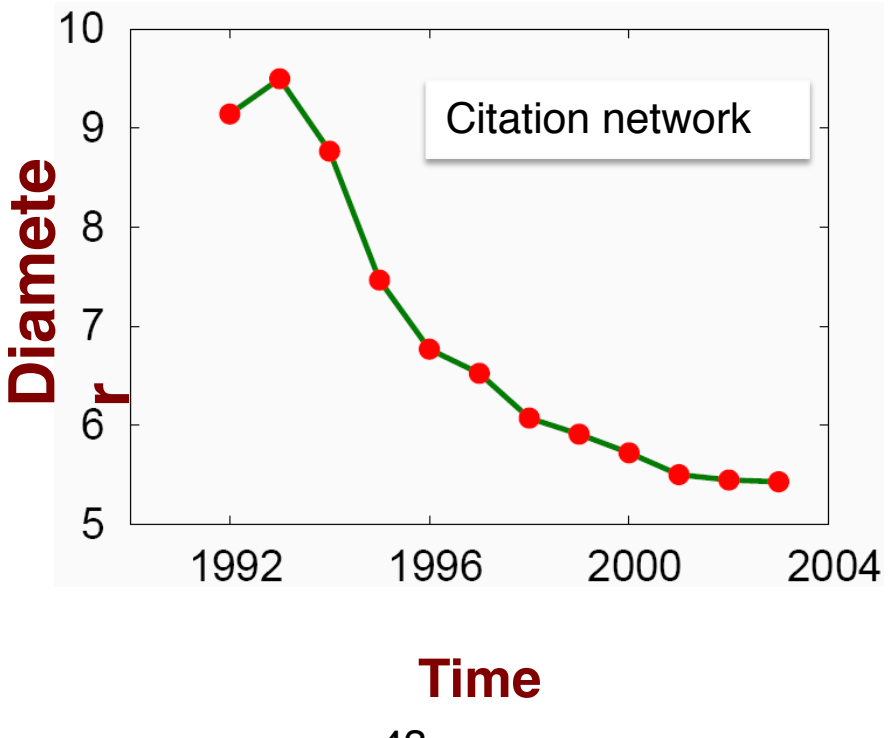
Source: www.kenedict.com

42

Shrinking Diameter

- **Q:** How does the **diameter** change, while the graph evolves with the addition of nodes and edges?
  - **Intuition:** the diameter should slowly grow (e.g., **log N, log log N**)
- **Diameter shrinks over time**

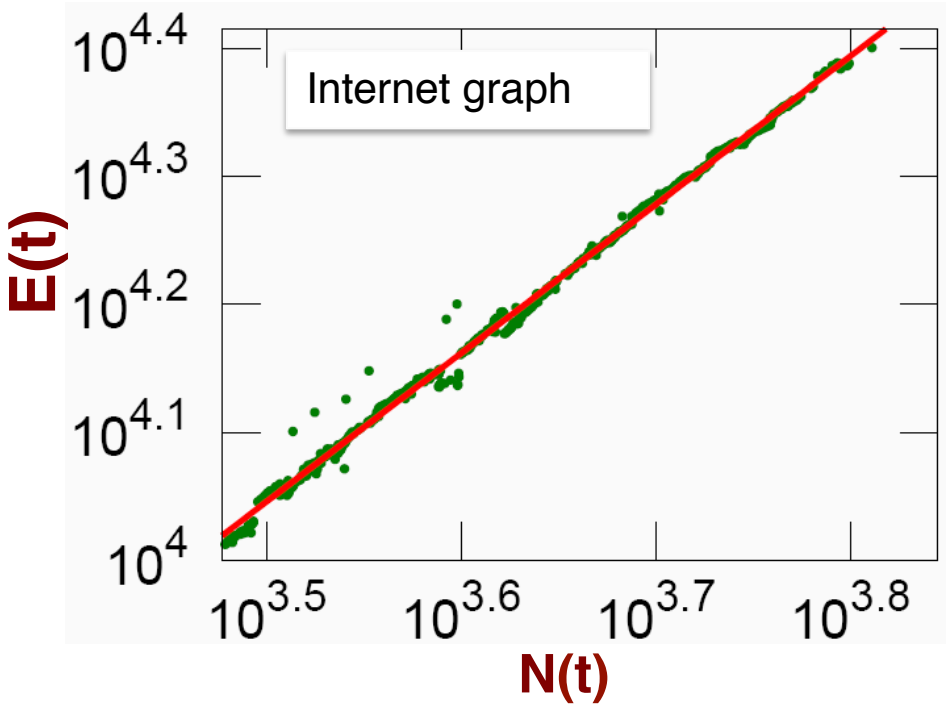


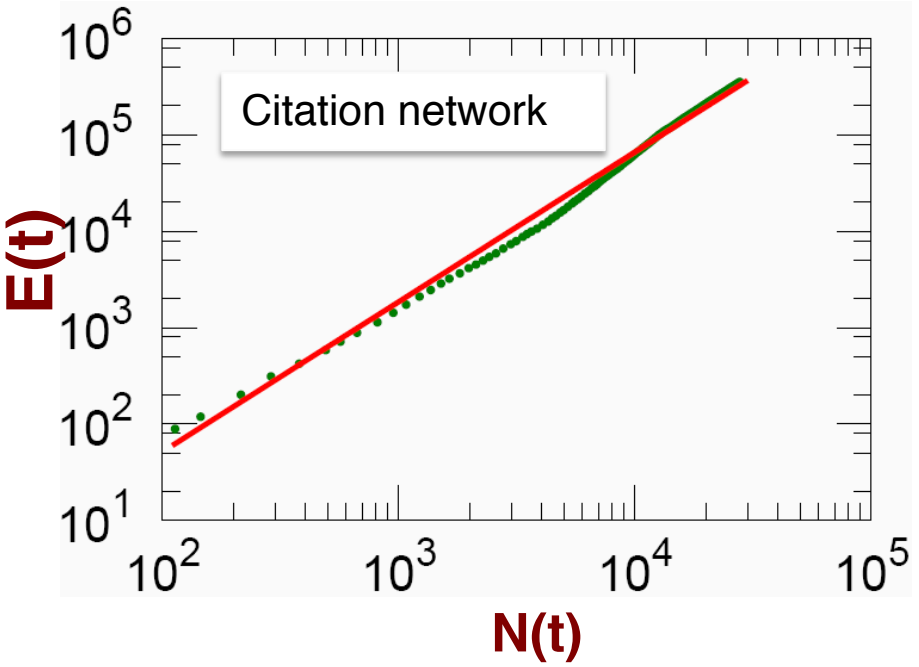


[Leskovec, 2009]

Densification Power Law

- **Q:** What is the relation between the number of nodes and edges over time?
- Networks become **denser** over time
  - **α** is the densification exponent (**1 <= α <= 2**) $E(t) \propto N(t)^a$





[Leskovec, 2009]

44

Outline

- Introduction and motivation
- Basic graph-theoretic concepts
- Properties of real-world networks
- Graph generating models

45

Network Evolution

Goal: Characterize, model and understand the structure of real networks

- How do real-world networks look like?
  1. **Empirical: statistical properties of networks** (e.g., degree distribution, diameter) **[Previous part]**

  2. **Generative models of network structure [Current part]**
     - Mechanisms that reproduce the underlying generative processes

Why do we Care?

- Creating models for real-world graphs is important for several reasons
  - Help us to **understand** and **reason** about the observed properties
  - Create **artificial data** for simulation purposes
  - **Predict** the evolution of networks
  - **Privacy preservation:** release the parameters of the generative model, instead of the network itself

47

What is a Network Model?

- Informally, it is a process (randomized or deterministic) for generating a graph
- Models of **static** graphs
  - **Input:** a set of parameter **Π** and the size of the graph **n**
  - **Output:** a graph **G(Π,n)**
- Models of **evolving** graphs
  - **Input:** a set of parameter **Π** and an initial graph $\mathbf{G_0}$
  - **Output:** a graph $\mathbf{G_t}$ for each time step **t**

Erdős–Rényi Random Graph Model

- Suppose that we want to generate a network with **n** nodes
- The $\mathbf{G_{n,\,p}}$ model:
  - Graph with **n** nodes and edge probability **p**
  - For each pair of nodes **(u, v)**, add the edge **(u, v)** **independently** with probability **p**
  - Family of graphs, in which a graph with **m** edges appears with probabilitv

$$p^m(1-p)^{\binom{n}{2}-m}$$

- The $\mathbf{G_{n,\,m}}$ model:
  - Select **m** edges uniformly at random

49

Degree Distribution of the ER Model (1/2)

- **Q:** Do Erdős–Rényi graphs look **realistic**?
- The degree distribution is **Binomial**
  - Let C_k denote the number of nodes with degree **k**

$$C_k = \binom{n-1}{k} p^k (1-p)^{n-1-k}$$

- What if **n → infinity** and we fix the expected degree = **c**?

$$\text{If } n \to \infty \text{ and } np \to c \text{ (with } c > 0\text{) then}$$

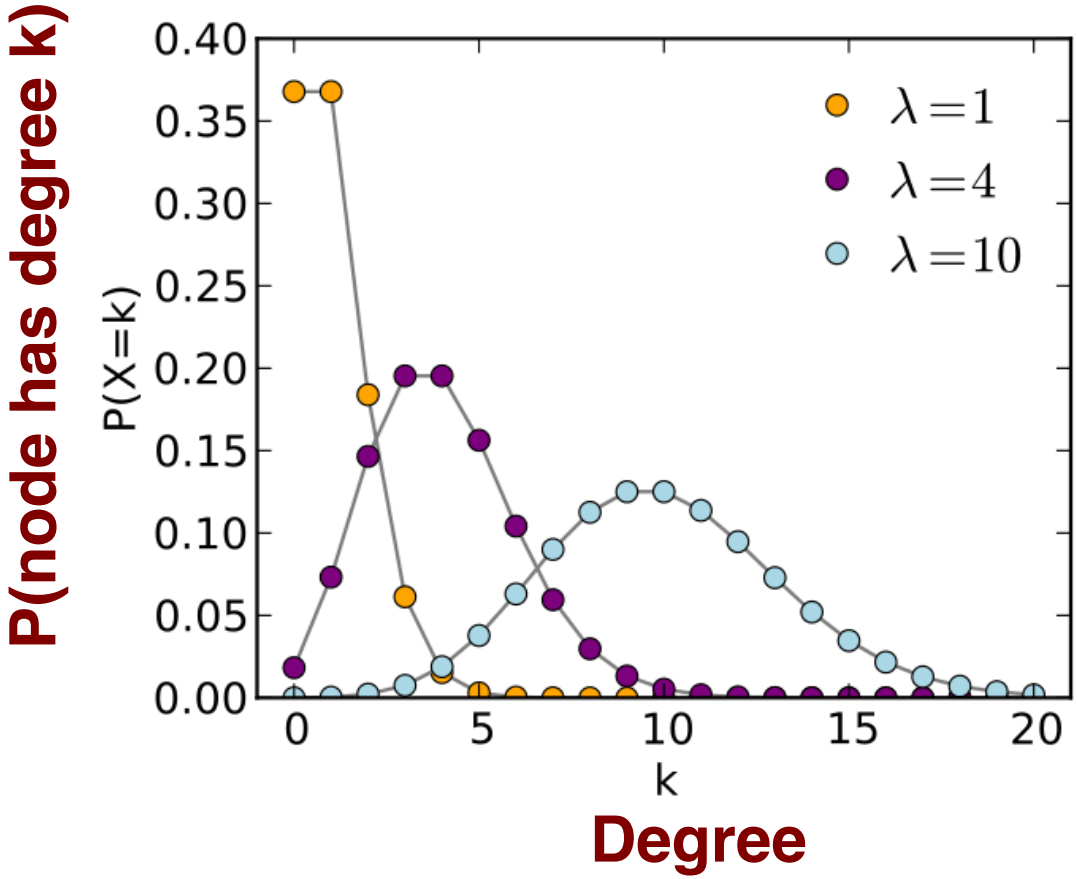
$$\frac{n!}{(n-k)!k!} p^k (1-p)^{n-k} \to e^{-c} \frac{c^k}{k!}$$

Poisson distribution

Degree Distribution of the ER Model (2/2)

Poisson distribution

$$\frac{\lambda^k e^{-\lambda}}{k!}$$



The degree distribution of ER random graph model is **not realistic** for real-world graphs

Source: Wikipedia

51

Preferential Attachment Model – General Idea

- Recall that real-world networks tend to have **power-law** (or in general heavy-tailed) degree distribution
- **Barabasi-Albert** (BA) model
  - Based on the idea of preferential attachment
- Intuition
  - Design a graph generating model that produces a small number of high degree nodes (hubs) and …
  - … also captures the long-tail (nodes with small degree)

Idea: Consider nodes that are more likely to connect to high-degree nodes

52

Barabasi-Albert Model (1/2)

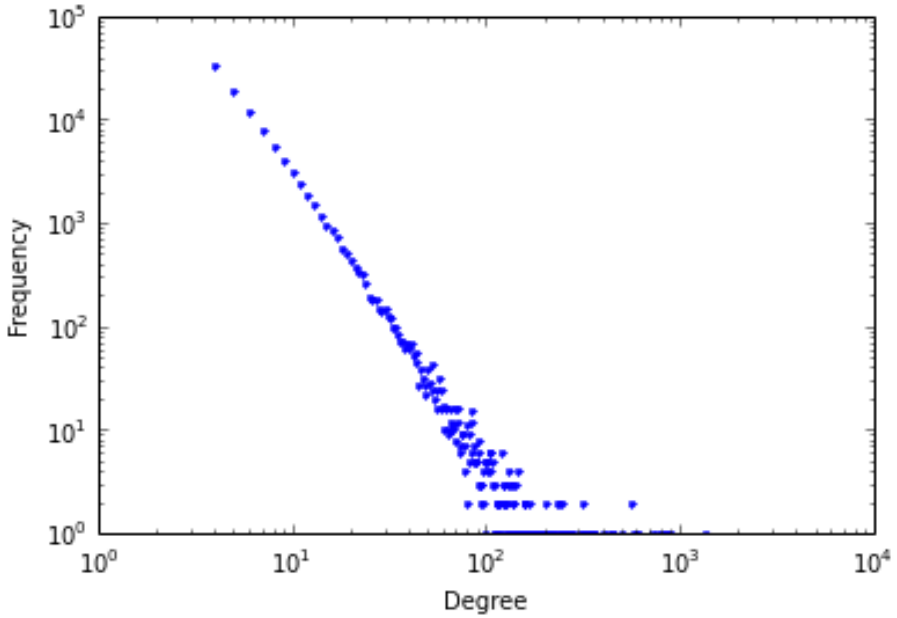
- The **Barabasi-Albert** model:
  - **Input:** some initial subgraph G_0 and a parameter **m** that corresponds to the number of edges per new node
  - The process:
    - The nodes arrive one at the time
    - Each new node connects to **m** existing nodes selected with probability proportional to their degree
    - Let **[d1, d2, …, dt]** be the degree sequence at time **t**. Then the node at **t+1** will be connected to node **i** with probability

$$p_i = \frac{d_i}{\sum_i d_i}$$

53

Barabasi-Albert Model (2/2)

- This phenomenon is also known as the **rich get richer** effect
  - E.g., a web page that already has many incoming hyperlinks is likely to get more in the future
- The BA model produces graphs with **power-law** degree distribution $C_k = k^{-\gamma}$, where $\gamma = 3$



- Barabasi-Albert graph
- **n = 100,000** nodes
- **m = 4**

The BA model holds for several real-world networks (flickr, Delicious, LinkedIn) [Leskovec et al., 2008]

54

Network Models and Temporal Evolution

- Most of the existing models (e.g., BA) consider that
  - The **number of edges** grows **linearly** with respect to the number of nodes
  - The **diameter increases** based on a factor of **log n** or **log log n**
- In real networks we have observed
  - **Densification power law**
  - **Shrinking diameter**

How to model the temporal evolution of real-world networks?

55

Kronecker Model of Graphs (1/4)

- Reminder: **Kronecker product** of matrices
  - **A = [a_{ij}]** an **n x m** matrix
  - **B = [b_{ij}]** an **p x q** matrix
  - Then **C = A ⊠ B** is defined as the **np x mq** matrix

$$\mathbf{C} = \mathbf{A} \otimes \mathbf{B} \begin{pmatrix} a_{1,1}\mathbf{B} & a_{1,2}\mathbf{B} & \cdots & a_{1,m}\mathbf{B} \\ a_{2,1}\mathbf{B} & a_{2,2}\mathbf{B} & \cdots & a_{2,m}\mathbf{B} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n,1}\mathbf{B} & a_{n,2}\mathbf{B} & \cdots & a_{n,m}\mathbf{B} \end{pmatrix}$$

- **Intuition:** repeat the Kronecker product between the adjacency matrix of an initial graph to get the final graph

[Leskovec et al., 2010]

Kronecker Model of Graphs (2/4)

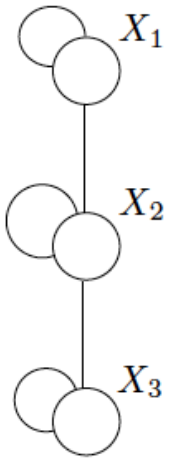
- **Kronecker** model:
  - Start by an initiator adjacency matrix $\mathbf{A}_1$ of size **p x p**
  - The Kronecker product of two graphs is defined as the Kronecker product of their adjacency matrices
  - The Kronecker graph after **k** iterations is defined as the graph with the following adjacency matrix

$$\mathbf{A}_k = \underbrace{\mathbf{A}_1 \otimes \mathbf{A}_1 \otimes \cdots \otimes \mathbf{A}_1}_{k \text{ iterations}} = \mathbf{A}_{k-1} \otimes \mathbf{A}_1$$

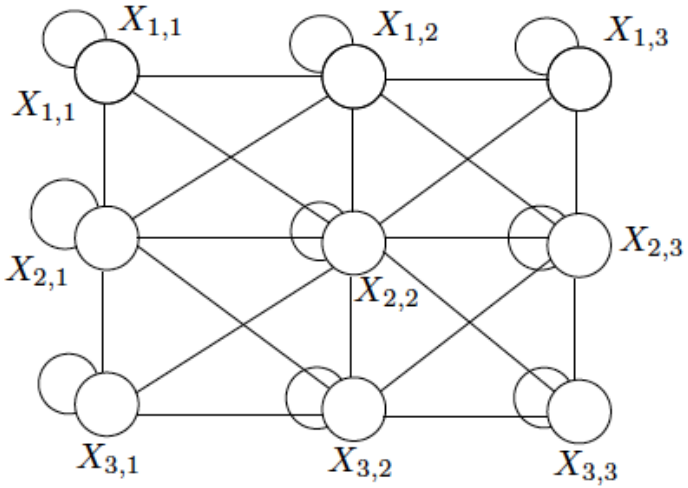
  - Each Kronecker multiplication exponentially increases the size of the graph

[Leskovec et al., 2010]

Kronecker Model of Graphs (3/4)



Graph G_1



Graph $G_2 = G_1 \boxtimes G_1$

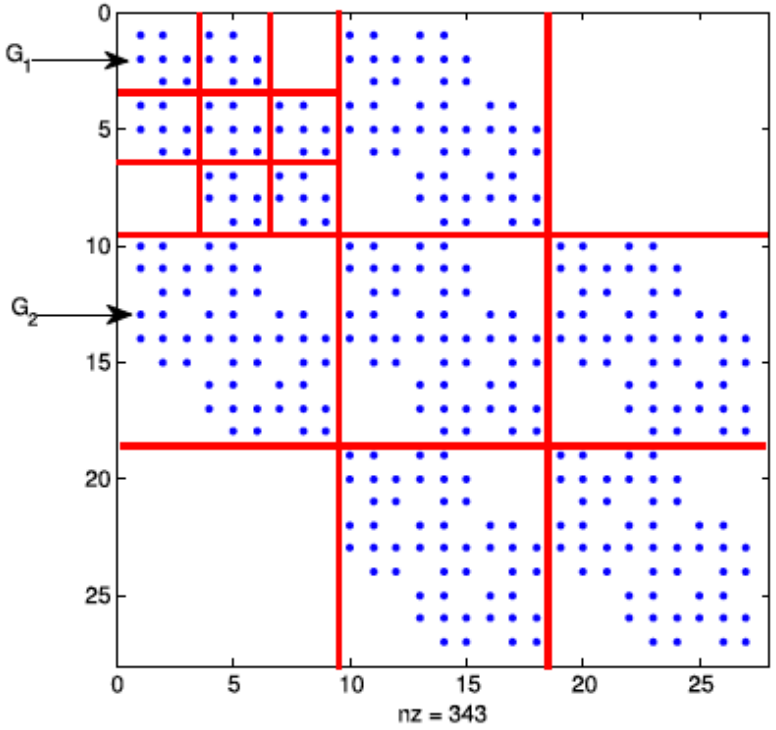
1	1	0
1	1	1
0	1	1

G_1	G_1	0
G_1	G_1	G_1
0	G_1	G_1

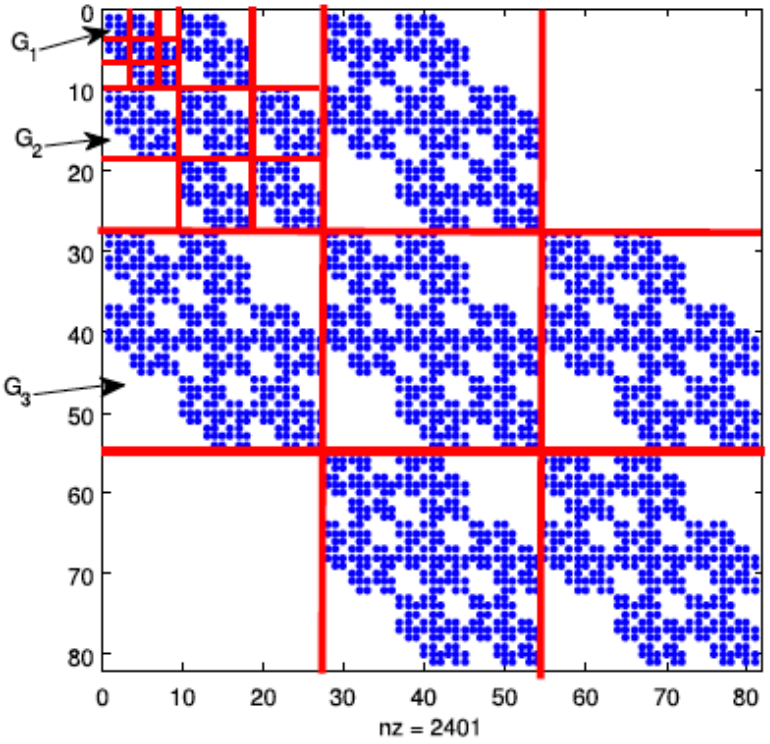
[Leskovec et al., 2010]

58

Kronecker Model of Graphs (4/4)



(α) $A(G_3) = A(G_2) \otimes A(G_1)$



(β) $A(G_4) = A(G_3) \otimes A(G_1)$

Intuition: Recursion and self-similarity

[Leskovec et al., 2010]

59

Stochastic Kronecker Model

- In practice, the **stochastic Kronecker graph** is used
  - Start by an initiator matrix **θ**

a	**b**
c	**d**

  - We obtain a graph with $\mathbf{n = 2^k}$ nodes by repeating **k** times the Kronecker product: $\mathbf{A_{k,\theta} = \theta \boxtimes \ldots \boxtimes \theta}$
  - Consider the value **(i, j)** of the matrix $\mathbf{A_{k,\theta}}$ as the probability of existence of the edge **(i, j)** (applying randomized rounding)
  - Typically, **2 x 2** initiator matrices produce good results

[Leskovec et al., 2010]

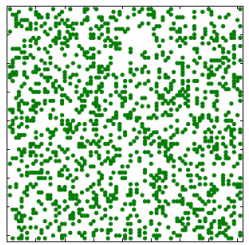
Generate Realistic Kronecker Graphs

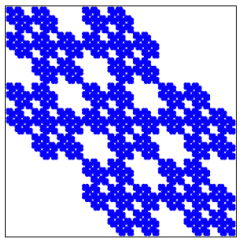
- Given a network **G**, how can we find a "good" initiator matrix **θ**, such that $\mathbf{A_G} \sim= \boldsymbol{\theta} \boxtimes \ldots \boxtimes \boldsymbol{\theta}$?
  - Fit the parameters **θ** of the model
  - Idea: use **maximum-likelihood estimation**

$$\arg\max_{\Theta} P(G|\Theta)$$

After Kronecker products

$$\arg\max_{G_1}\left(\quad \right)$$





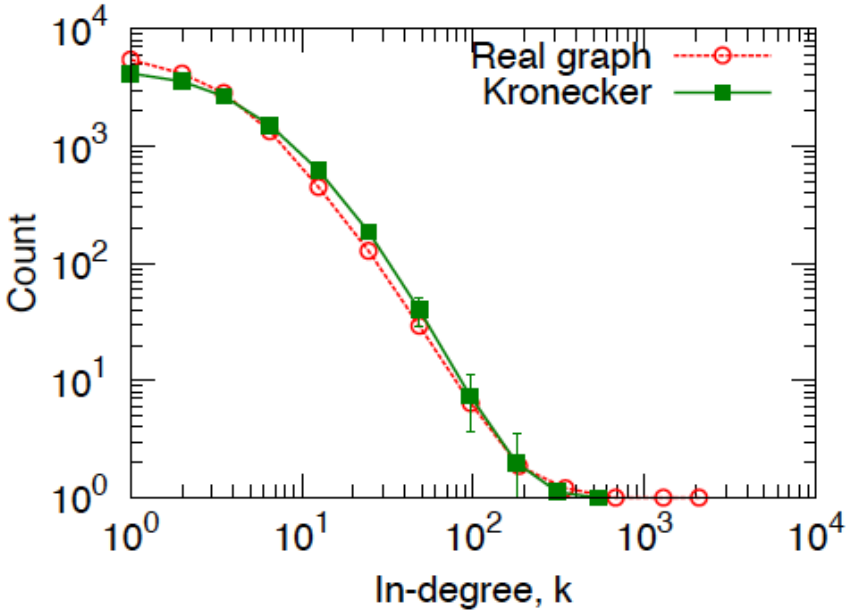
[Leskovec et al., 2010]

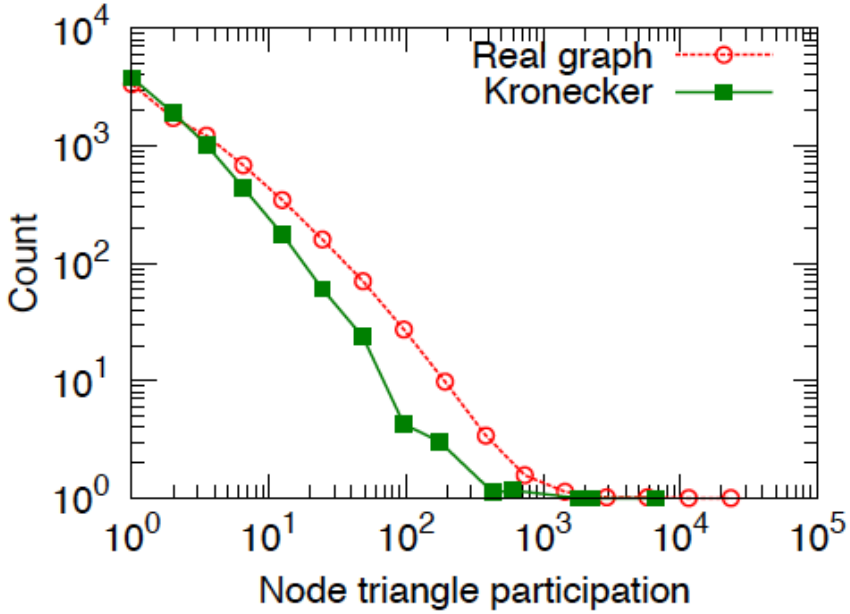
Properties of Kronecker Model

- The Kronecker (stochastic) graph model is able to reproduce a plethora of properties
  - Power-law degree distribution
  - Small diameter
  - Shrinking diameter
  - Densification power-law
  - Triangle participation
  - …

[Leskovec et al., 2010]

Example: Fitting Kronecker Model to a Graph





Blog-to-Blog network

[Leskovec et al., 2010]

63

References

- J. Leskovec. Modeling Large Social and Information Networks. Tutorial at ICML, 2009.
- J. McAuley. Data Mining and Predictive Analytics, UCSD, 2015.
- D. Easley and J. Kleinberg. Networks, Crowds, and Markets: Reasoning About a Highly Connected World. Cambridge University Press, 2010.
- J. Leskovec, D. Chakrabarti, J. Kleinberg, C. Faloutsos, Z. Ghahramani. Kronecker Graphs: An approach to modeling networks. JMLR, 2010.