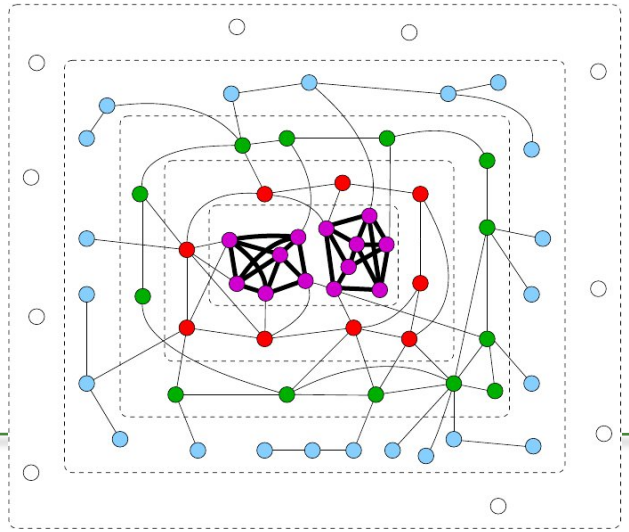
Graph Mining



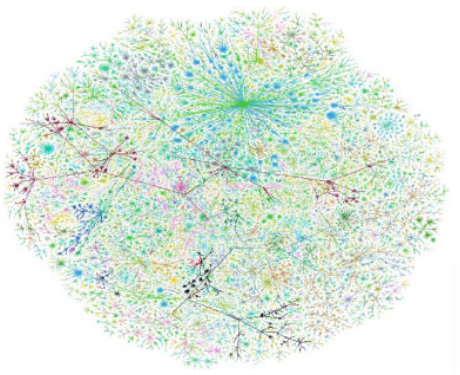
Michalis Vazirgiannis
LIX @ Ecole Polytechnique
http://www.lix.polytechnique.fr/dascim/
Google Scholar: https://bit.ly/2rwmvQU

November 2019

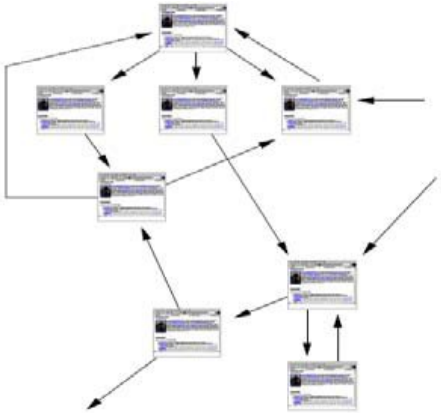
Outline

- Motivating Graph Mining and basic notions
- Graph Exploration and Generation
  - Degree distribution – power laws
  - Graph generators
- Machine Learning for Graphs
  - Community detection
  - Supervised learning with Graph Kernels

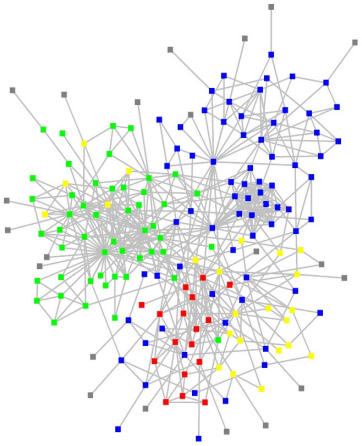
Networks are Everywhere



(a) Internet

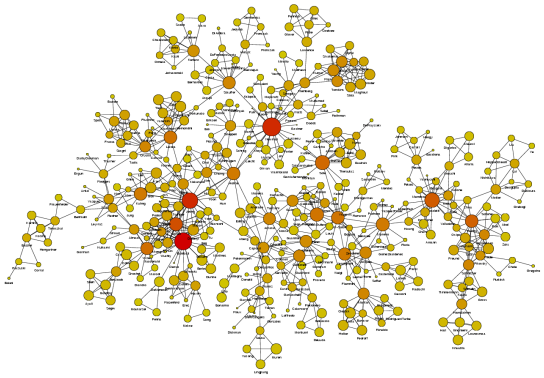


(b) World Wide Web

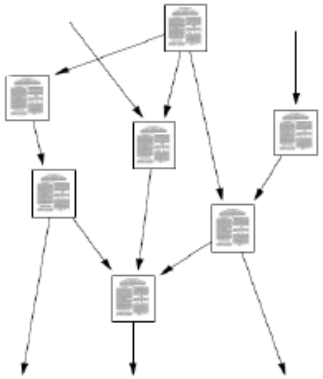


(c) Email network

(d) Social network



(e) Collaboration network

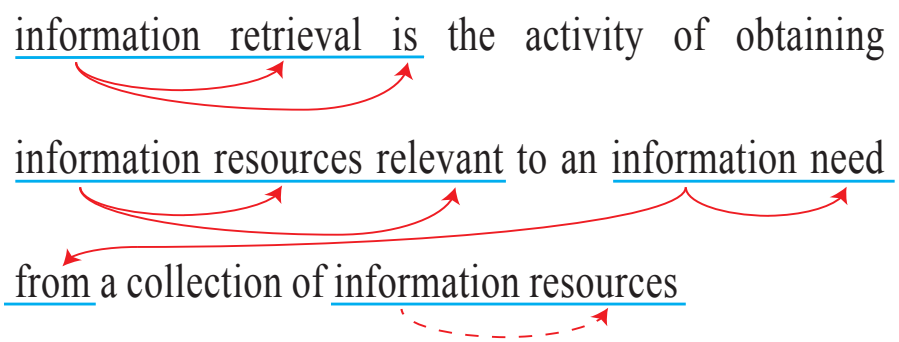


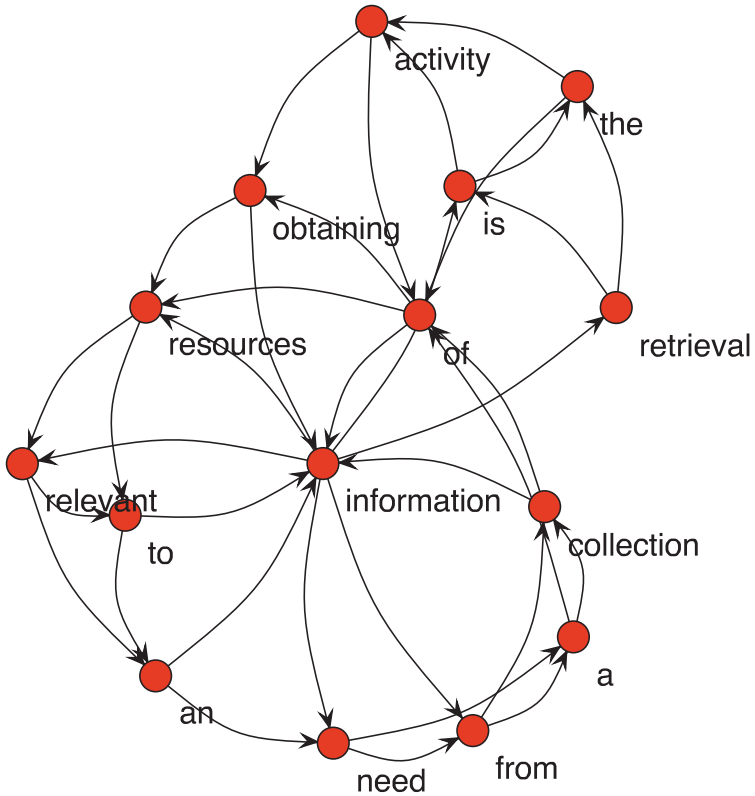
(f) Citation network

Graphs are ubiquitous!

- **Technological networks:**
  - Internet
  - Telephone networks
  - Power grid
  - Road, airline and rail networks
- **Information networks:**
  - World Wide Web
  - Blog networks
  - Citation networks
- **Social networks:**
  - Collaboration networks
  - Organizational networks
  - Communication networks
- **Biological networks:**
  - Networks from Neuroscience
  - Protein-protein interaction networks
  - Gene regulatory networks
  - Food webs
- **Software networks:**
  - Call graphs
  - Software module/component interaction networks

Even representing text - Graph-of-word





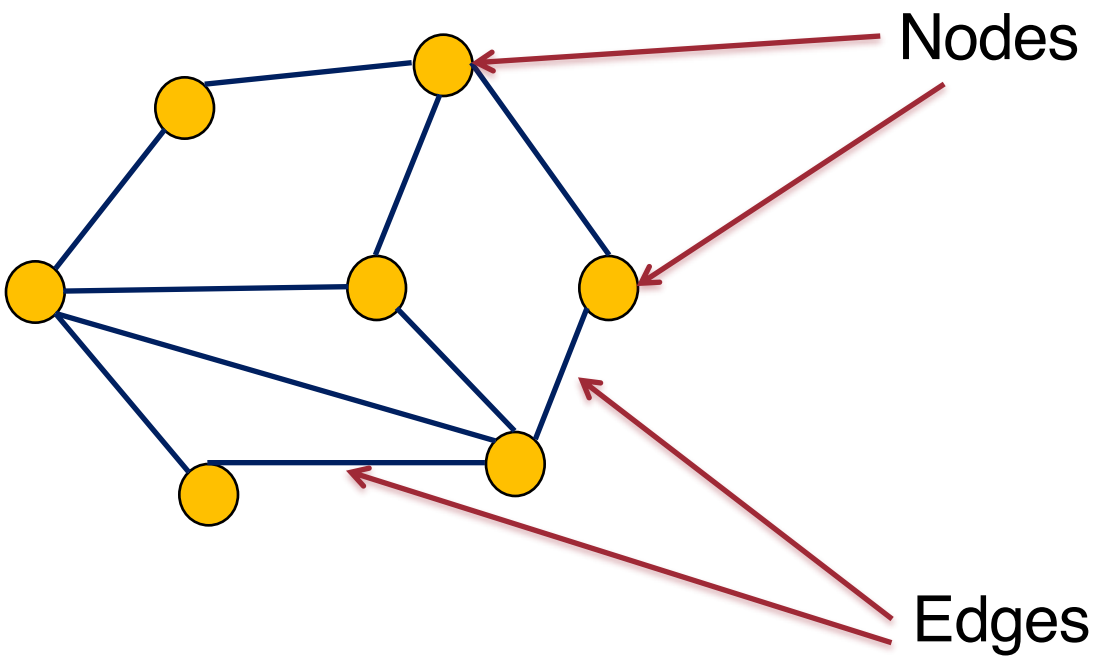
"Graph of word approach for ad-hoc information retrieval", F. Rousseau, M. Vazirgiannis, Best paper mention award ACM CIKM 2013

Elements of Learning from Graph data

- **Graph models/ graph generators** graph generators (erdos reyni, preferential attachment, kronecker graphs)
- **Node base metrics**: - Ranking algorithms (Pagerank), Ranking evaluation measures (Kendal Tau, NDCG),
- **Graph exploration/preprocessing**: degree distributions, visualization
- **Supervised learning for graphs**: link prediction, graph kernels, graph classification
- **Unsupervised learning:** clustering, community mining, degeneracy.
- **Learning theory in graphs:** model ensembling/selection…

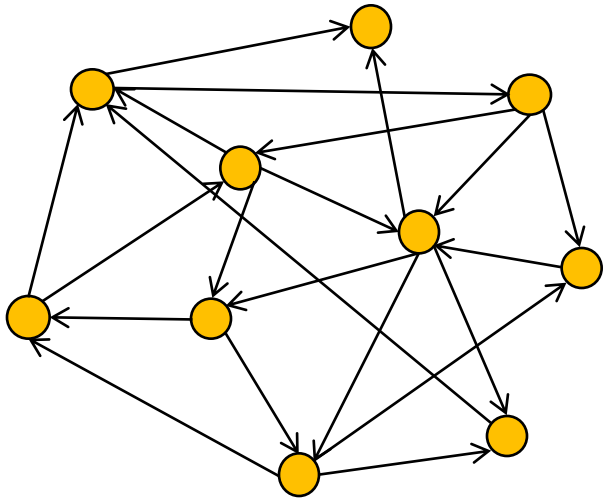
Graphs and Networks

- Graphs allow for modeling dependencies

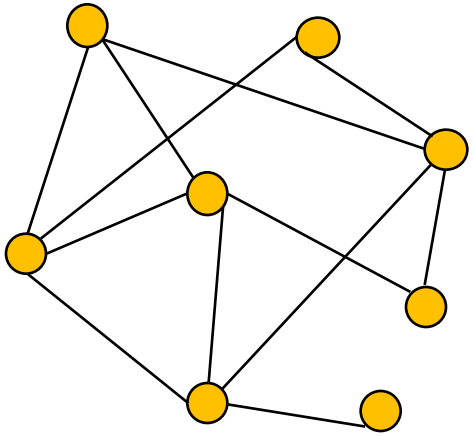


Basic Graph Definitions

- 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



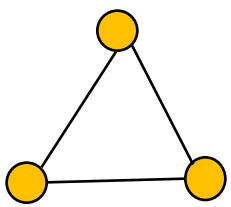
Undirected

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

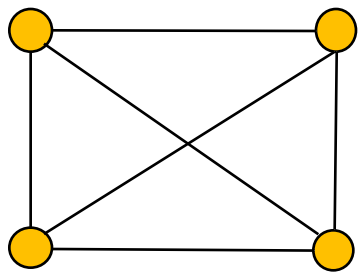
Degree: $d(i) = d_{in}(i) = d_{out}(i)$

Complete Graph

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



Complete graph with 3 nodes: triangle



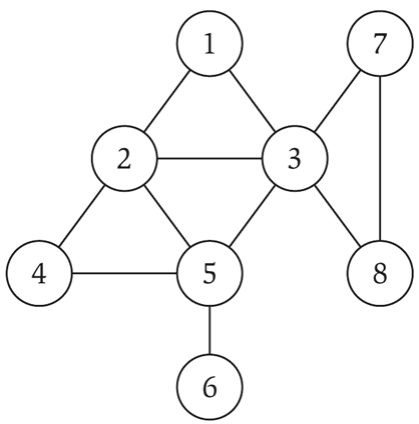
Complete graph with 4 nodes

- 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
  - $\mathbf{W_{ij} > 0}$, if **i** and **j** are connected
  - $\mathbf{W_{ij} = 0}$, if **i** and **j** are not connected
  - In case of unweighted graphs, $\mathbf{W_{ij} = 1}$, if **(i, j)** is an edge of the graph
  - Space proportional to $\mathbf{n^2}$



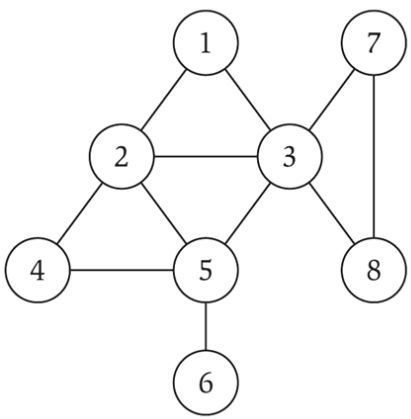
Undirected graph

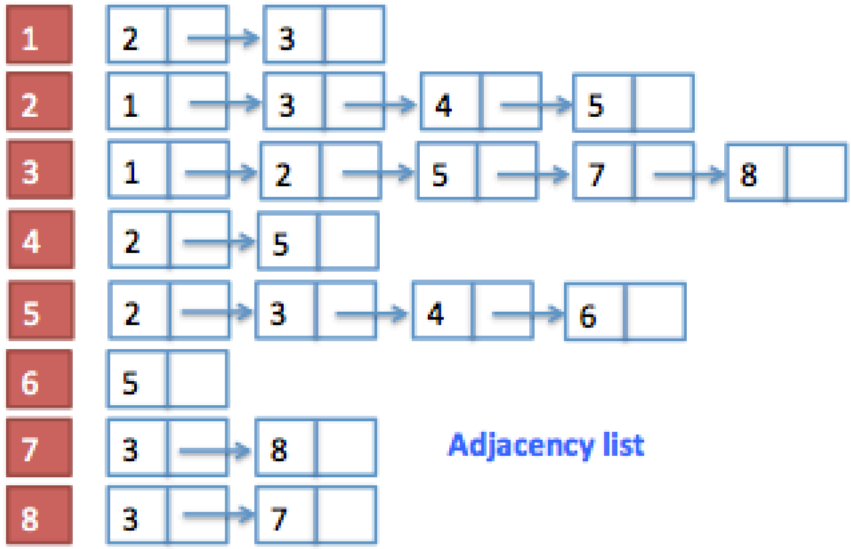
0	1	1	0	0	0	0	0
1	0	1	1	1	0	0	0
1	1	0	0	1	0	1	1
0	1	0	0	1	0	0	0
0	1	1	1	0	1	0	0
0	0	0	0	1	0	0	0
0	0	1	0	0	0	0	1
0	0	1	0	0	0	1	0

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 **$\Theta(m+n)$**
  - Checking if **(i, j)** is an edge takes **$O(d_i)$** time

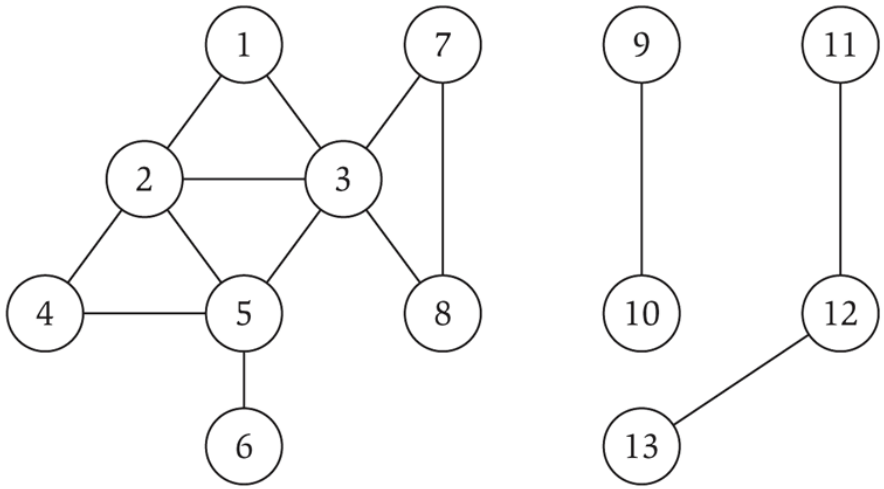




Paths and Connectivity in Graphs

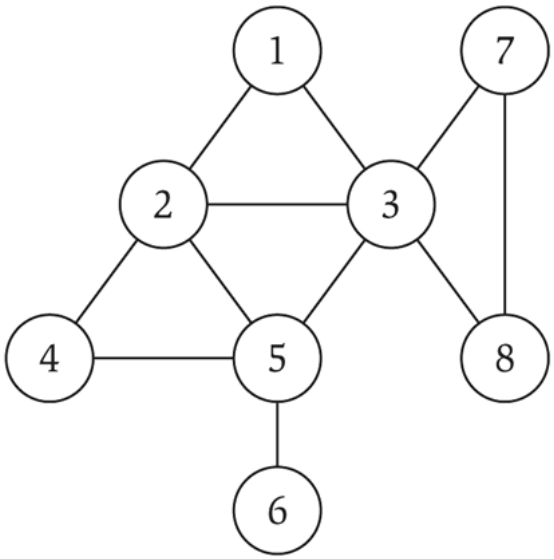
- **Definition:** A path in an undirected graph **G=(V,E)** is a sequence of nodes $v_1, v_2, \ldots, v_k$ with the property that each consecutive pair v_{i-1}, v_i is joined by an edge in **E**

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



Cycles in Graphs

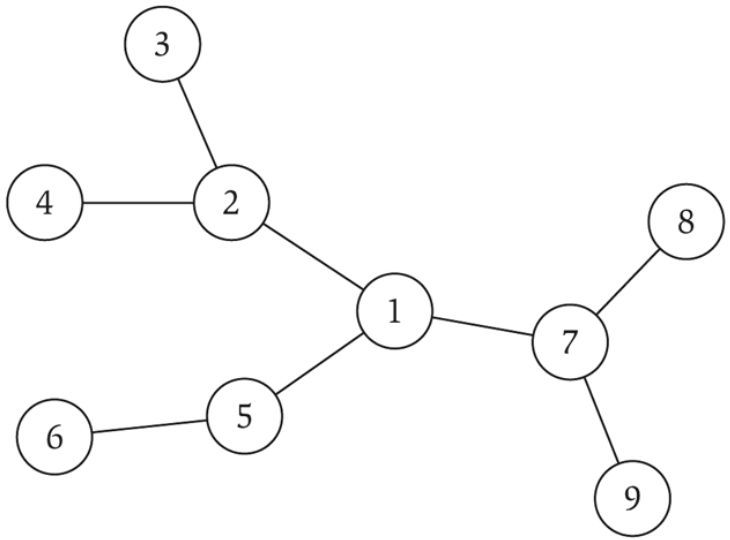
- **Definition:** A cycle is a path $v_1, v_2, \ldots, v_k$ in which $v_1 = v_k$, $k > 2$ and the first $k-1$ nodes are all distinct



Cycle C = 1 – 2 – 4 – 5 – 3 – 1

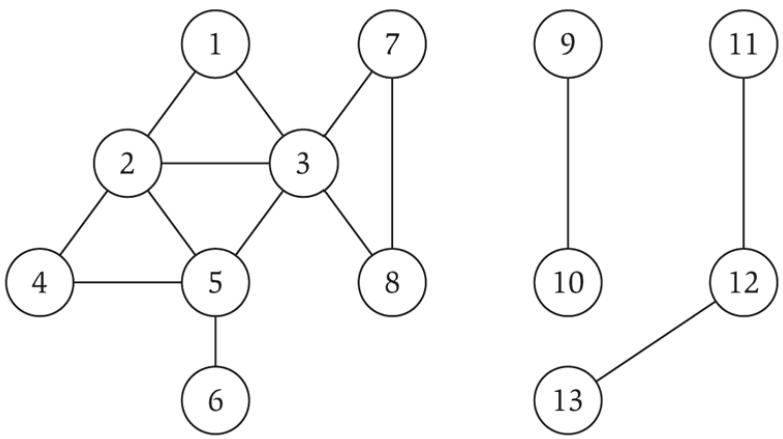
Trees

- **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



Connected Components

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



Graph with 3 connected components

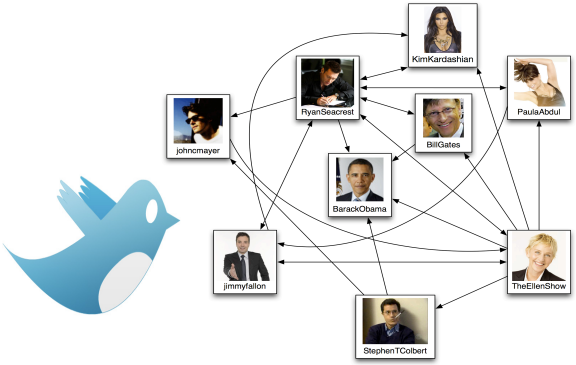
Connected component containing node 1: {1, 2, 3, 4, 5, 6, 7, 8}

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

A: Apply BFS

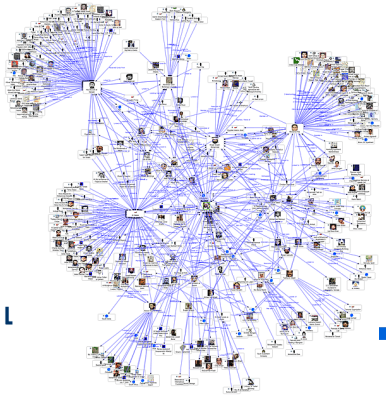
Connectivity in Directed Graphs

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

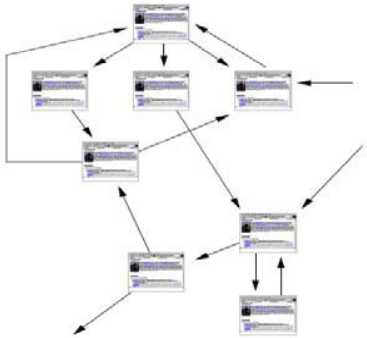


Twitter

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

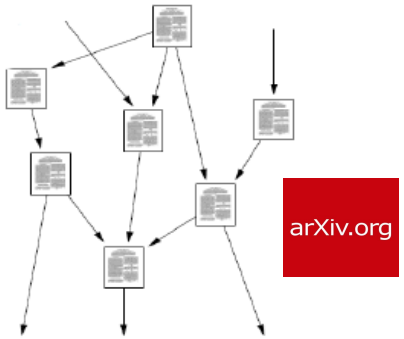


Online Social Networks



Web Graph

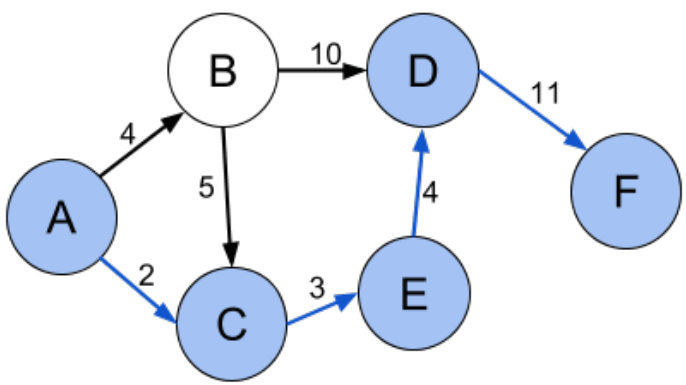
Wikipedia



Citation Graph

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

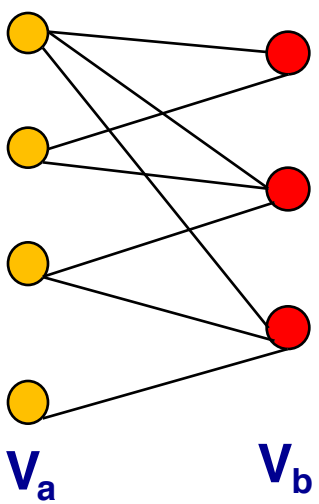


Shortest path (A, C, E, D, F) between vertices A and F in the weighted directed graph

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



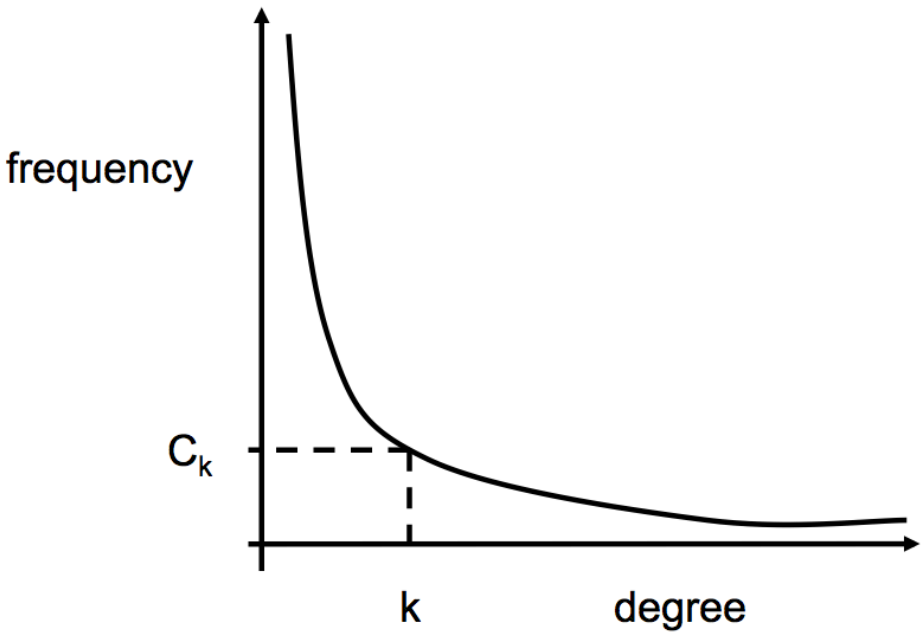
- Strong modeling capabilities and many real-world applications
- E.g., **Collaborative filtering** in recommender systems
  - Model the customer-product space using a bipartite graph (who-purchased-what)
  - If a user A has purchased the same product with a user B, then it is more likely to purchase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

Properties of Real-World Graphs

- 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

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**

Power-law Degree Distribution

- Let C_k = number of nodes with degree k

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

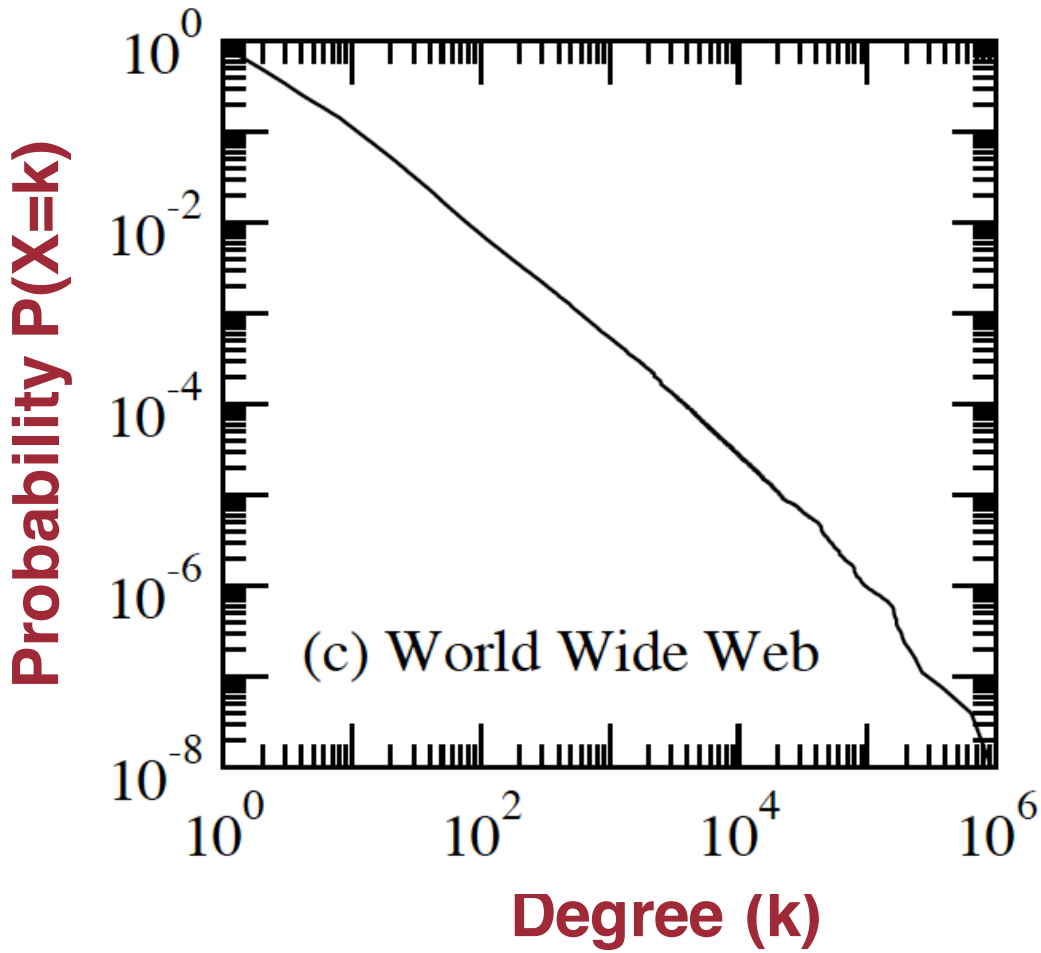
with $\gamma > 1$ and c a constant

- How to recognize a power-law distribution?

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

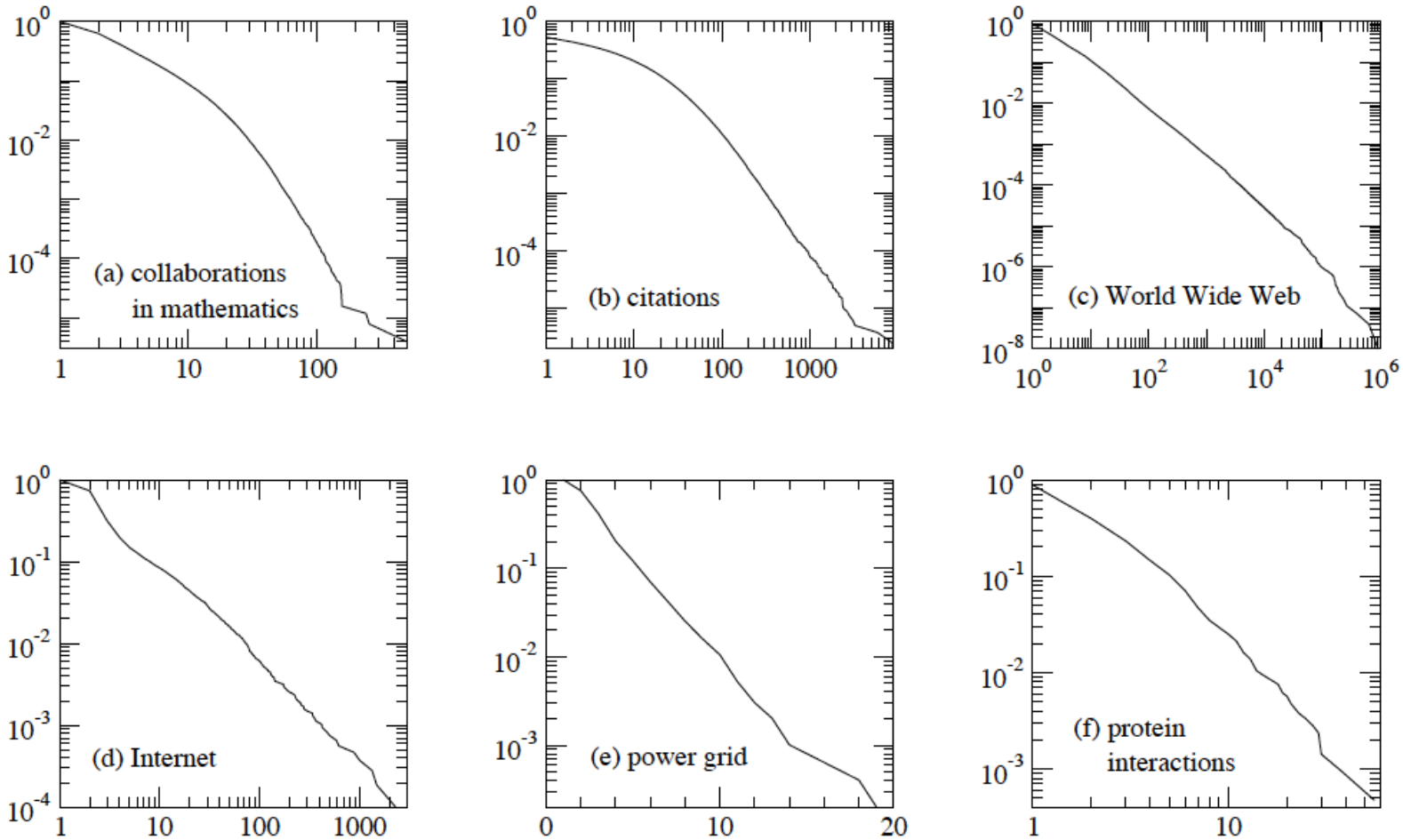
  - Plotting $\ln C_k$ versus $\ln k$ gives a straight line with slope $-\gamma \ln k$

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



[Newman, 2003]

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



Cumulative degree distribution for six different networks [Newman 2003]

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]

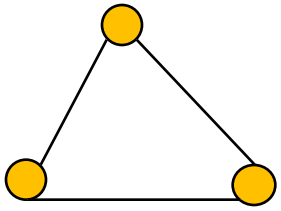
Summary – Degrees in Real Networks

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

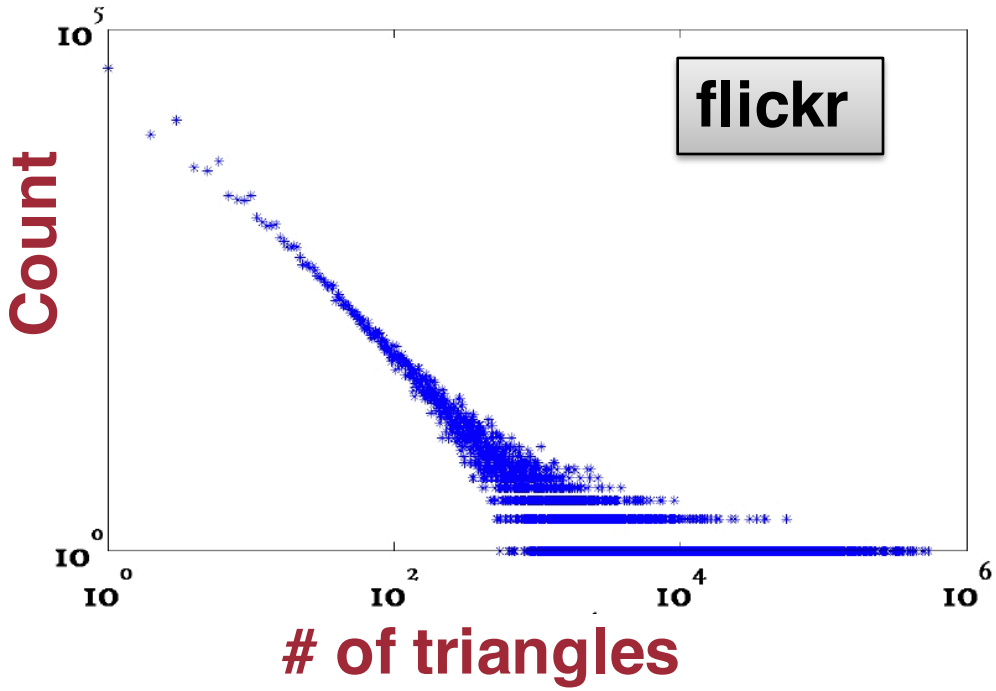
$$\lim_{x \to \infty} \frac{Pr(X > x)}{e^{-\epsilon x}} = \infty$$

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

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]

Clustering Coefficient

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

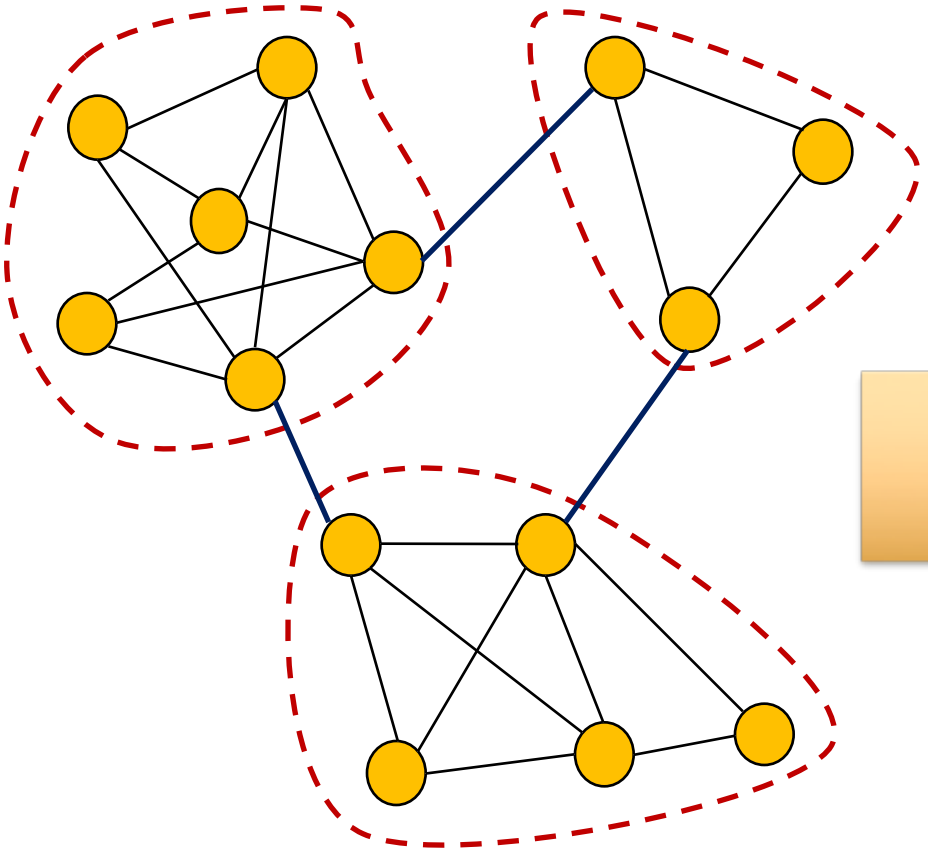
  T(G) = 3 x # of triangles in G / # of connected triplets

- 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

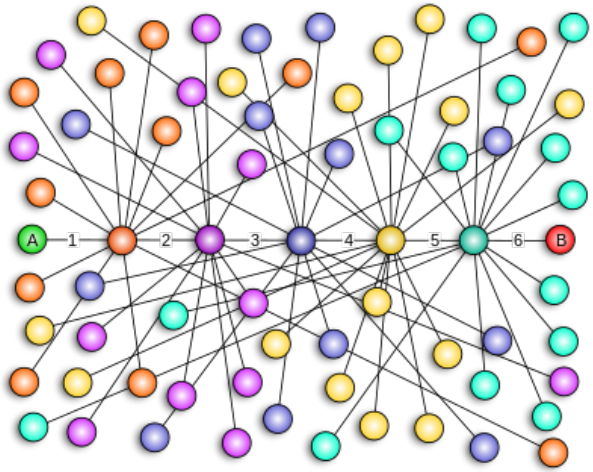


Example graph with three communities

- Will be covered later on in detail

Small-world Phenomenon (1/4)

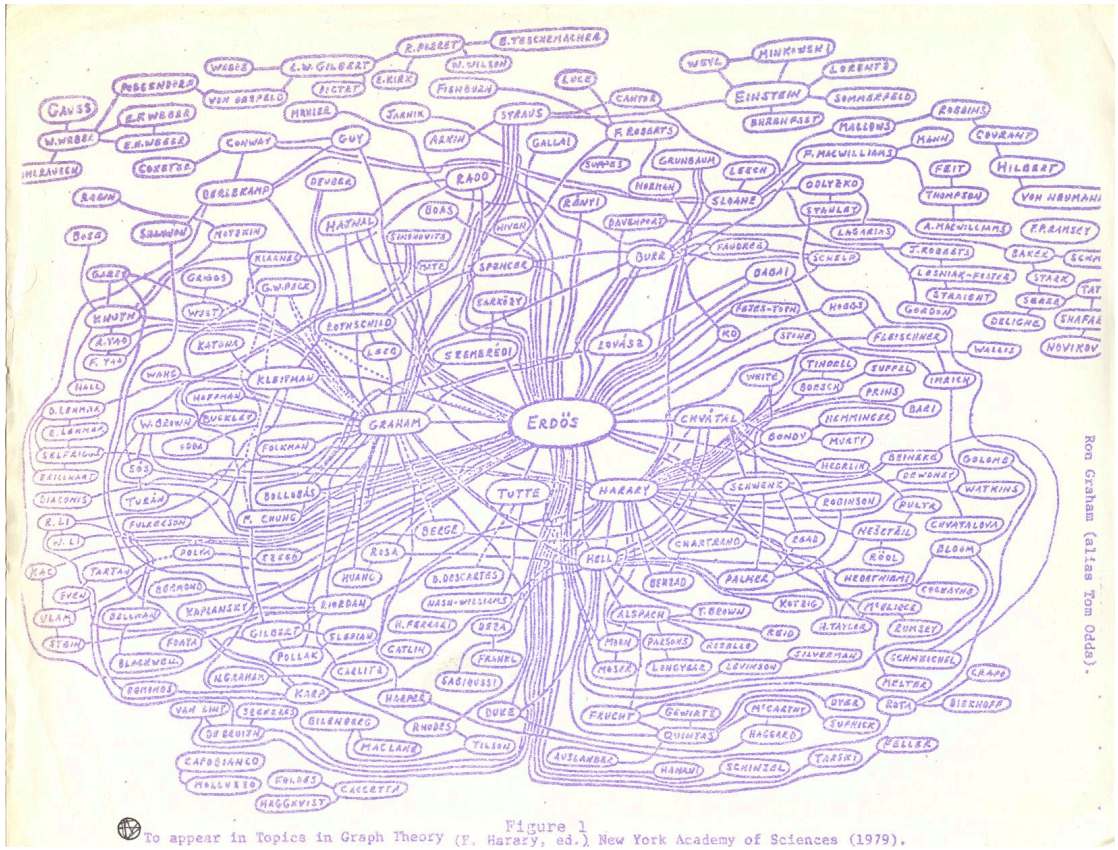
- Six degrees of separation
  - Experiment done by sociologist Stanley Milgram (1960's)
  - Randomly selected people in Nebraska were asked to send letters to Boston, by contacting somebody with whom they had direct connection
  1. People either sent the letter directly to the recipient
  2. Or to somebody 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



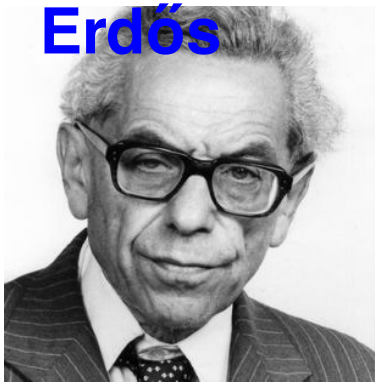
[McAuley, 2015]

Small-world Phenomenon (3/4)

- The small-world phenomenon appears in various network settings



Paul Erdős



Source: UCSD

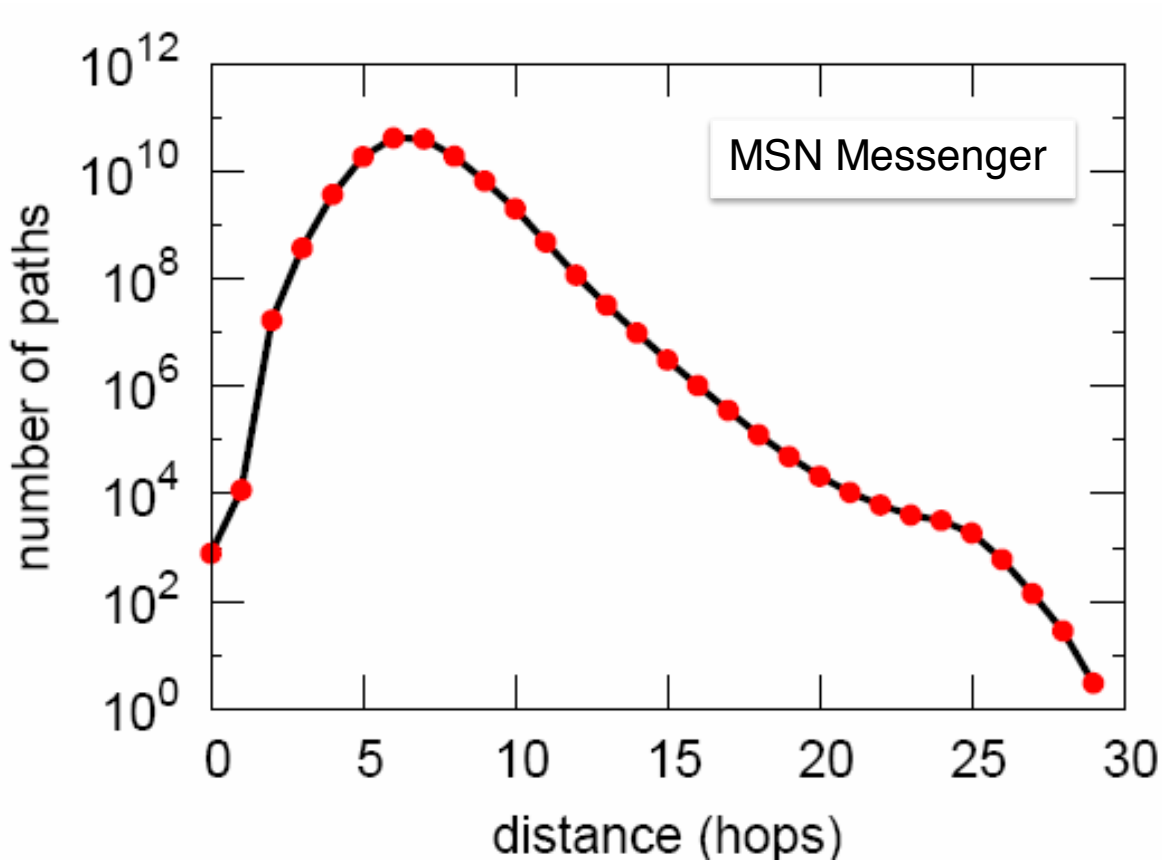
Source: physicsbuzz.physicscentral.com

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

30

Small-world Phenomenon (4/4)

- The small-world phenomenon appears in various network settings



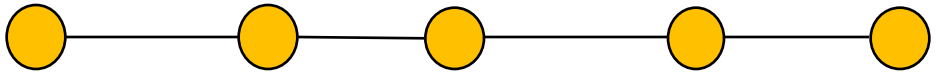
- **Average path length** is **6.6**
- 90% of the nodes are reachable in less than 8 steps

- **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



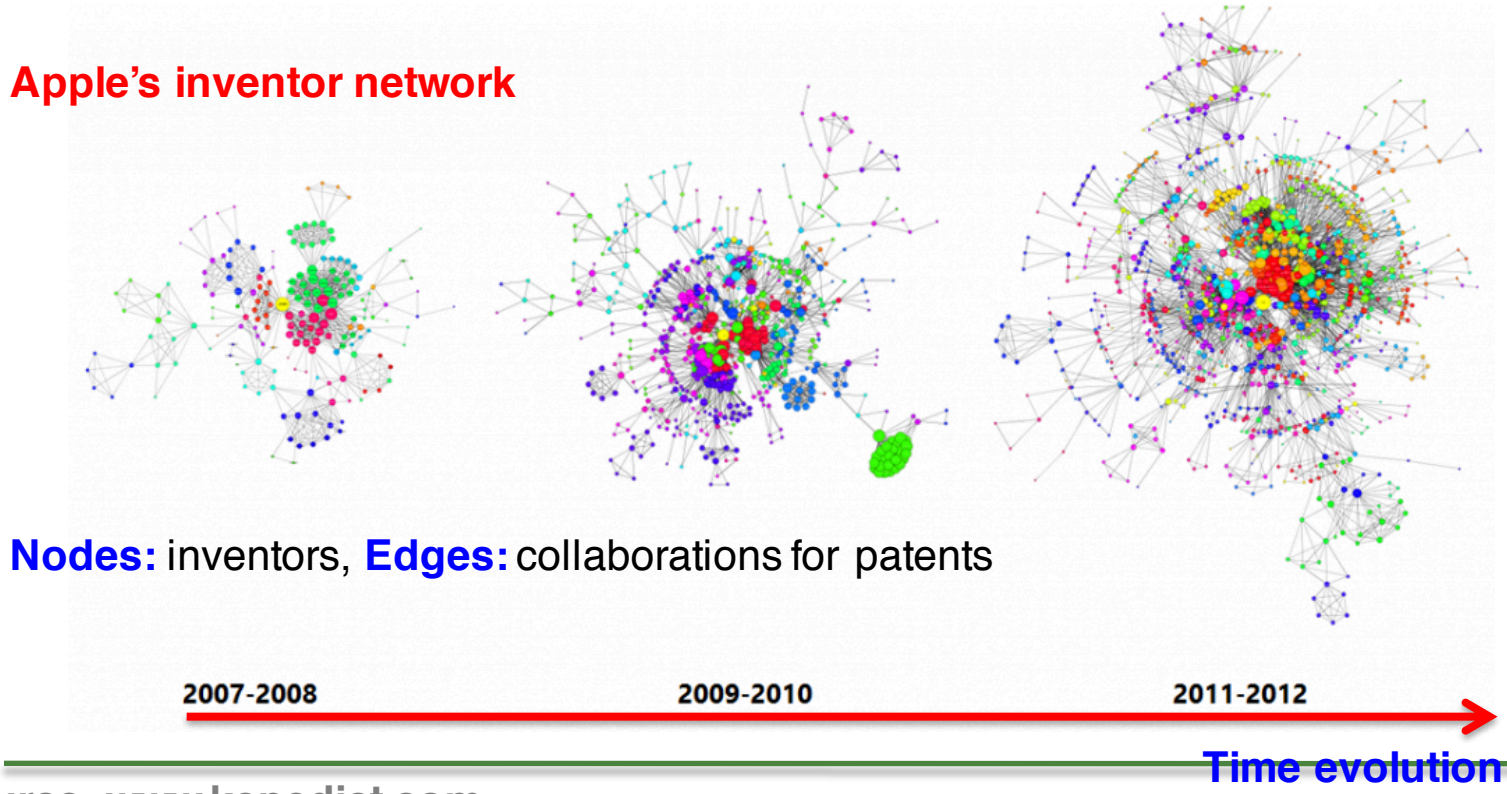
- In practice, we use the **effective diameter**
  - Upper bound of the shortest path over 90% of the pairs of nodes

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

[Leskovec, 2009]

Network Evolution

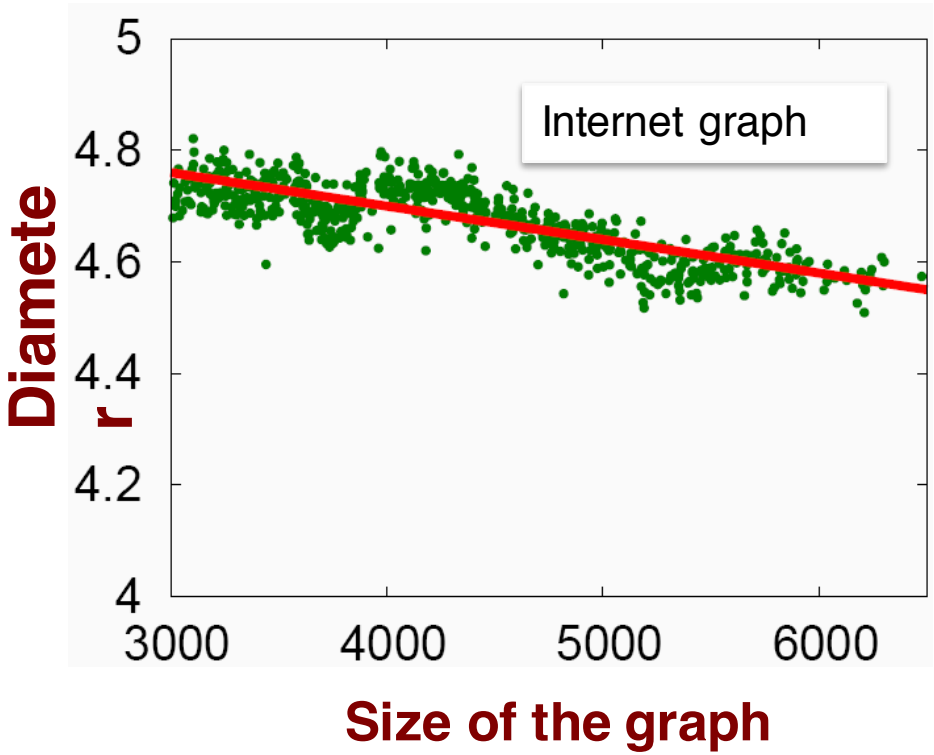
- Real-world networks are not static, but they evolve over time
  - New nodes/edges are added and/or deleted
  - We are interested in making predictions about the structure of the network

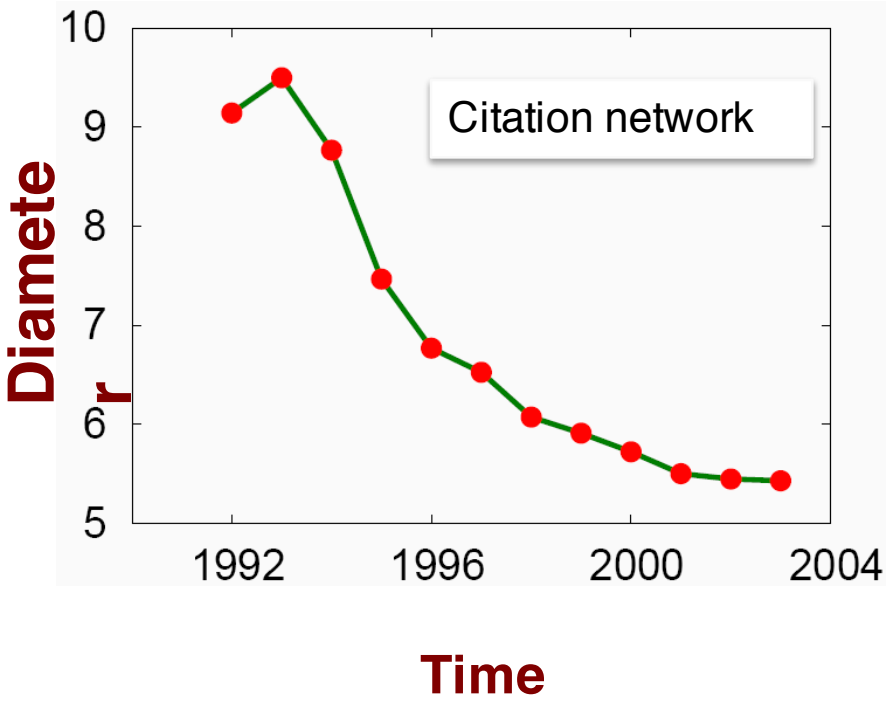


Source: www.kenedict.com

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]

Outline

- Motivating Graph Mining and basic notions
- Graph Exploration and Generation
  - Degree distribution – power laws
  - **Graph generators**
- Machine Learning for Graphs
  - Community detection
  - Supervised learning with Graph Kernels

Graph Generators - 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

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 $G(\Pi,n)$
- Models of **evolving** graphs
  - **Input:** a set of parameter Π and an initial graph G_0
  - **Output:** a graph 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 probability

- The $\mathbf{G_{n,\,m}}$ model: $p^m(1-p)^{\binom{n}{2}-m}$
  - Select **m** edges uniformly at random

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**?

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

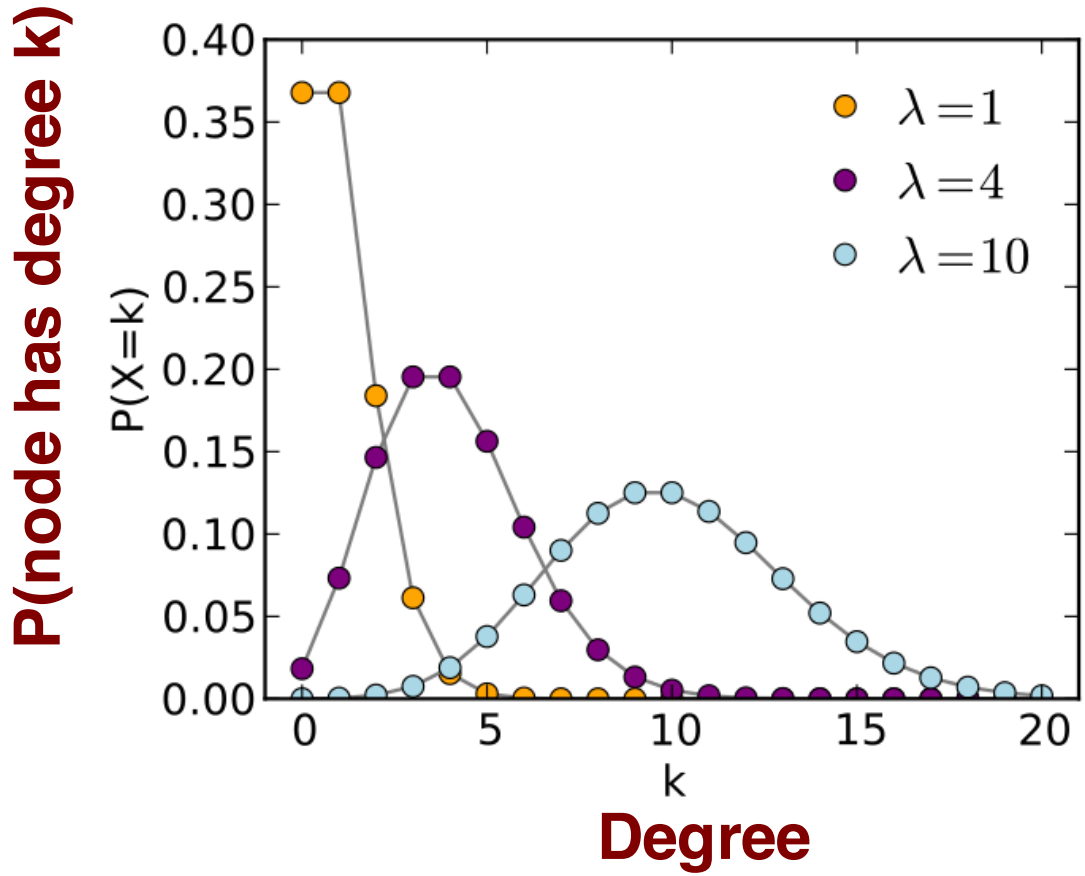
$$\frac{n!}{(n-k)!k!} p^k (1-p)^{n-k} \to \boxed{e^{-c} \frac{c^k}{k!}}$$

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

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

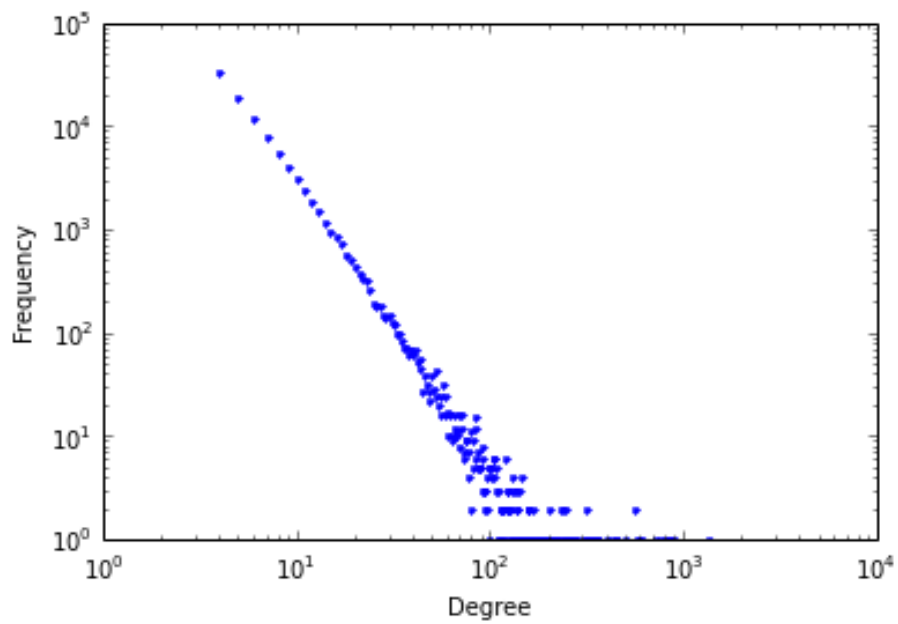
Barabasi-Albert Model (1/2)

- 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

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

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]**

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?

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)

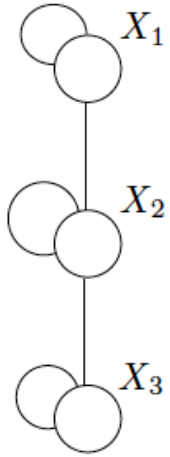
- **Kronecker** model:
  - Start by an initiator adjacency matrix $\mathbf{A_1}$ of size **p x p**
  - The Kronecker product of two graphs is defined as the Kronecker product of their adjacency matrices
  - The Kronecker graph after **k** iterations is defined as the graph with the following adjacency matrix

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

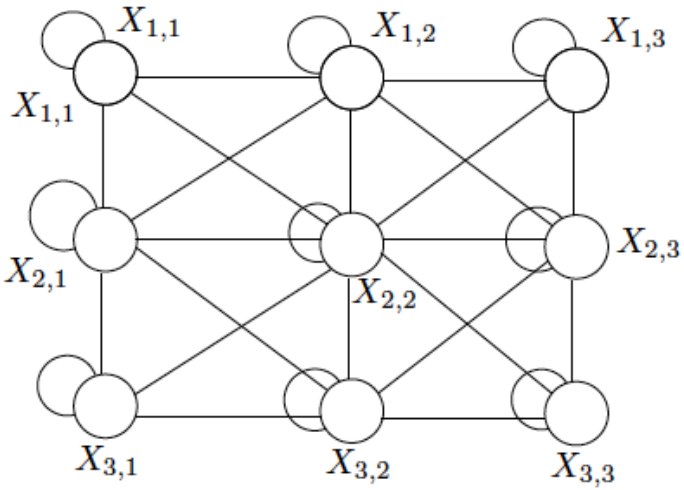
  - Each Kronecker multiplication exponentially increases the size of the graph

[Leskovec et al., 2010]

Kronecker Model of Graphs (3/4)



Graph G_1



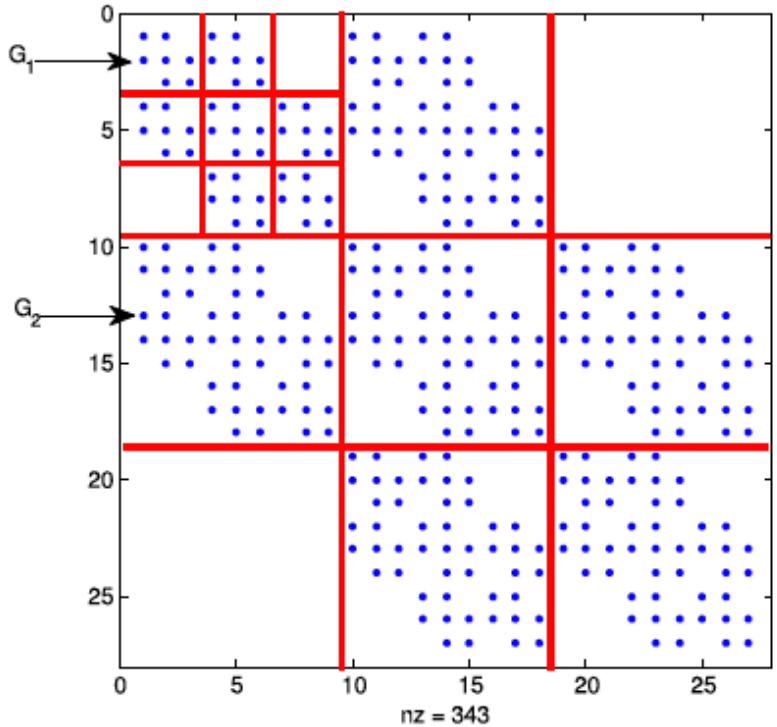
Graph $G_2 = G_1 \boxtimes G_1$

1	1	0
1	1	1
0	1	1

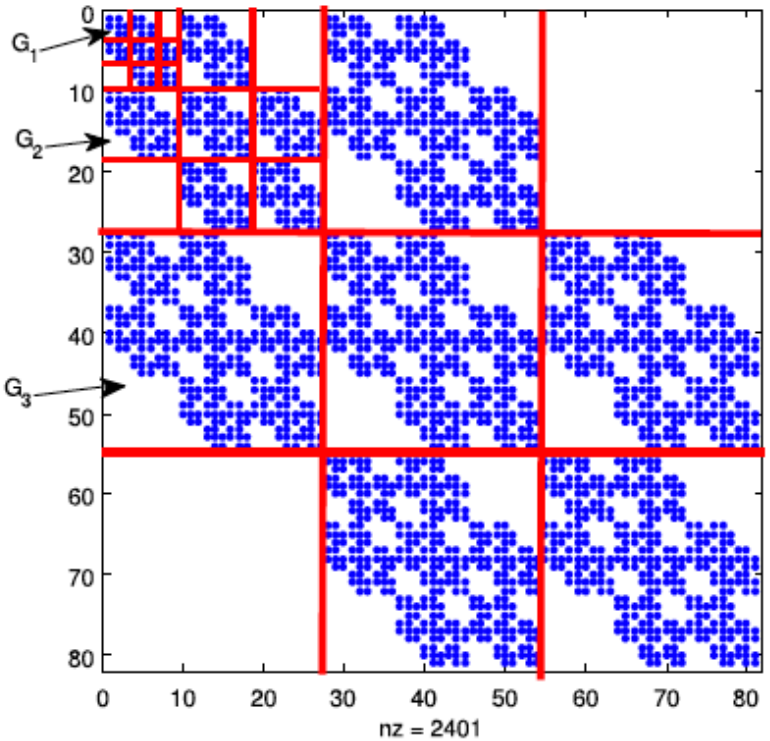
G_1	G_1	0
G_1	G_1	G_1
0	G_1	G_1

[Leskovec et al., 2010]

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]

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

[Leskovec et al., 2010]

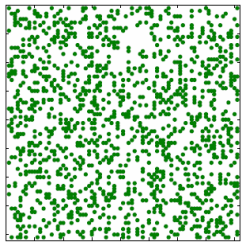
Generate Realistic Kronecker Graphs

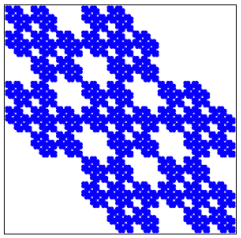
- Given a network **G**, how can we find a "good" initiator matrix **θ**, such that $\mathbf{A_G} \sim= \boldsymbol{\theta} \boxtimes \ldots \boxtimes \boldsymbol{\theta}$?
  - Fit the parameters **θ** of the model
  - Idea: use **maximum-likelihood estimation**

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

After Kronecker products

$$\arg\max_{G_1}\Big(\quad \mathbf{I} \quad \Big)$$





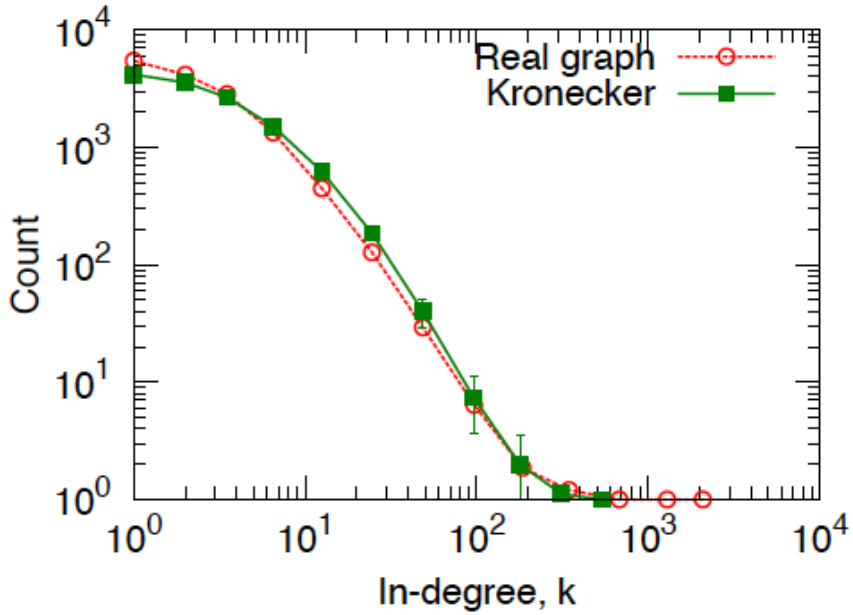
[Leskovec et al., 2010]

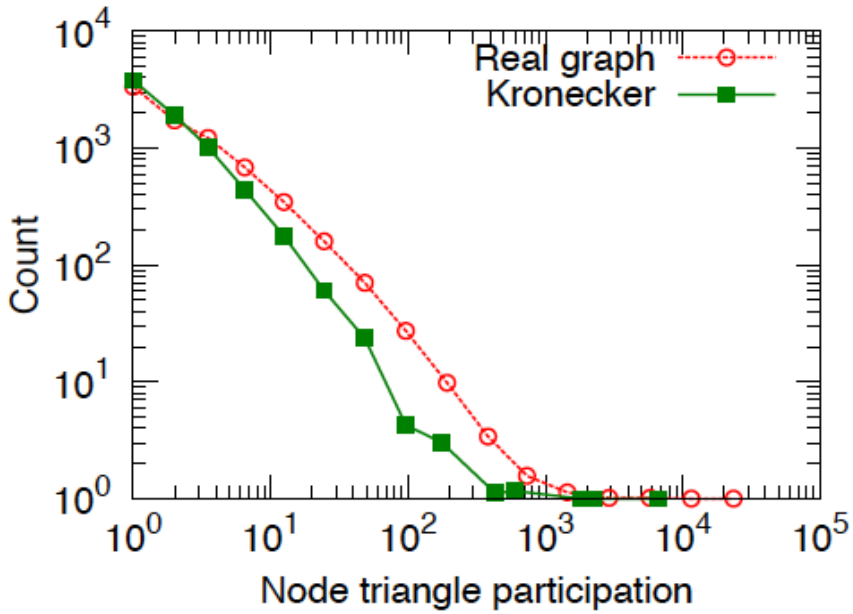
Properties of Kronecker Model

- 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]

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.

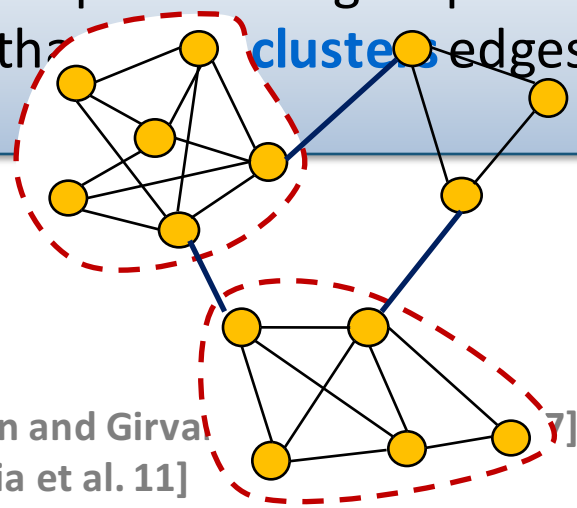
2. Community evaluation measures

3.

Basics

- The notion of **community structure** captures the tendency of nodes to be organized into modules (communities, clusters, groups)
  - Members within a community are **more similar** among each other
- Typically, the communities in graphs (networks) correspond to **densely connected** entities (nodes)

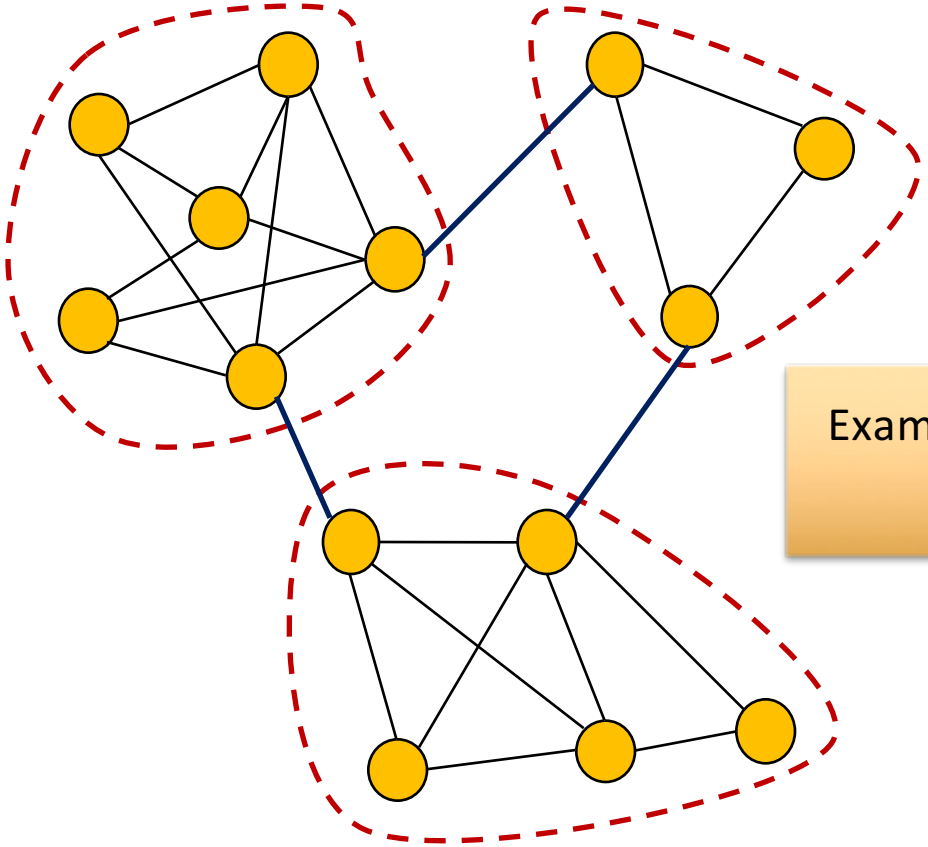
A community corresponds to a group of nodes with more **intra-cluster** edges than **inter-cluster** edges



Example graph with three communities

[Newman '03], [Newman and Girvan [illegible]7], [Fortunato '10], [Danon et al. '05], [Coscia et al. 11]

Schematic representation of communities



Example graph with three communities

Community detection in graphs

- How can we extract the inherent communities of graphs?
- Typically, a two-step approach
  1. Specify a **quality measure** (evaluation measure, objective function) that quantifies the desired properties of communities
  2. Apply **algorithmic techniques** to assign the nodes of graph into communities, optimizing the objective function
- Several measures for quantifying the quality of communities have been proposed
- They mostly consider that communities are set of nodes with many edges between them and few connections with nodes of different communities
  - Many possible ways to formalize it

Community evaluation measures

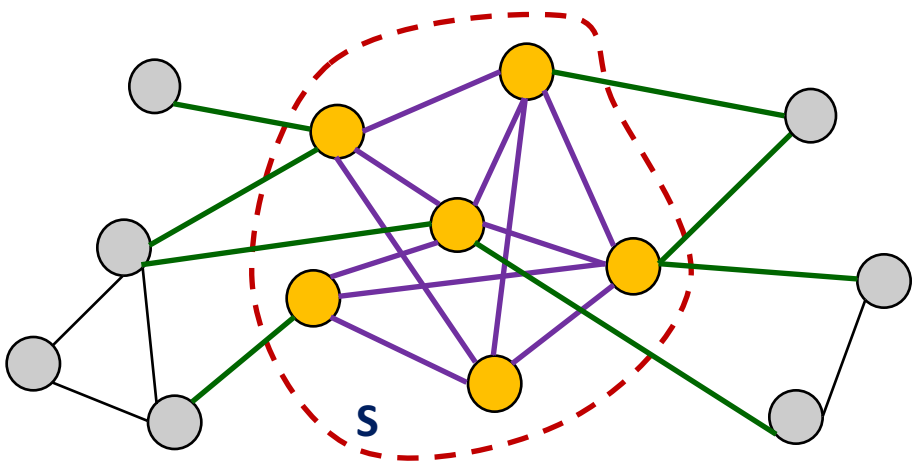
- Focus on
  - Intra-cluster edge density (# of edges within community),
  - Inter-cluster edge density (# of edges across communities)
  - Both two criteria

- We group the community evaluation measures according to
  - Evaluation based on **internal** connectivity
  - Evaluation based on **external** connectivity
  - Evaluation based on **internal and external** connectivity
  - Evaluation based on **network model**

[Leskovec et al. '10], [Yang and Leskovec '12], [Fortunato '10]

Notation

- **G = (V, E)** is an undirected graph, **|V| = n**, **|E| = m**
- **S** is the set of nodes in the cluster
- n_s **= |S|** is the number of nodes in **S**
- m_s is the number of edges in **S**, $m_s = \left|\{(u,v): u \in S, v \in S\}\right|$
- c_s is the number of edges on the boundary of **S**, $c_s = \left|\{(u,v): u \in S, v \notin S\}\right|$
- d_u is the degree of node ***u***
- ***f* (S)** represent the clustering quality of set ***S***



Nodes in **S** (n_s)

Edges in **S** (m_s)

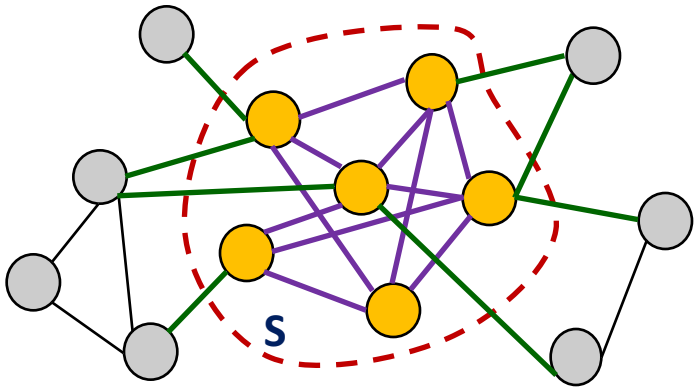
Edges in boundary of **S** (c_s)

Evaluation based on internal and external connectivity

- **Conductance** [Chung '97]

$$f(S) = \frac{c_s}{2m_s + c_s}$$

Measures the fraction of total edge volume that points outside **S**



- **Normalized cut** [Shi and Malic '00]

$$f(S) = \frac{c_s}{2m_s + c_s} + \frac{c_s}{2(m - m_s) + c_s}$$

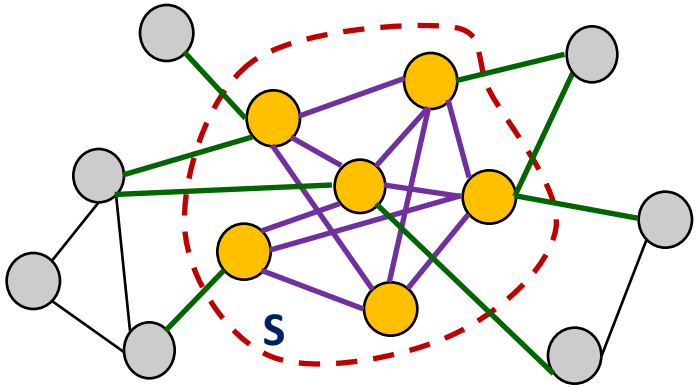
Measures the fraction of total edge volume that points outside **S** normalized by the size of **S**

Evaluation based on internal connectivity

- **Triangle participation ratio (TPR)** [Yang and Leskovec '12]

$$f(S) = \frac{\left|\left\{u : u \in S, \left\{(v,w) : v,w \in S, (u,v) \in E, (u,w) \in E, (v,w) \in E\right\} \neq \varnothing\right\}\right|}{n_s}$$

Fraction of nodes in **S** that belong to a triangle



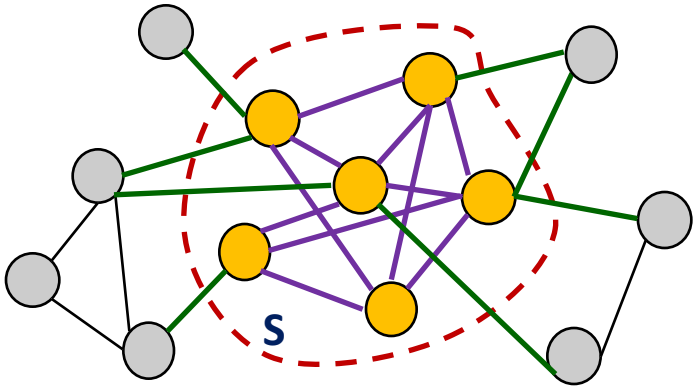
Evaluation based on network model

- **Modularity** [Newman and Girvan '04], [Newman '06]

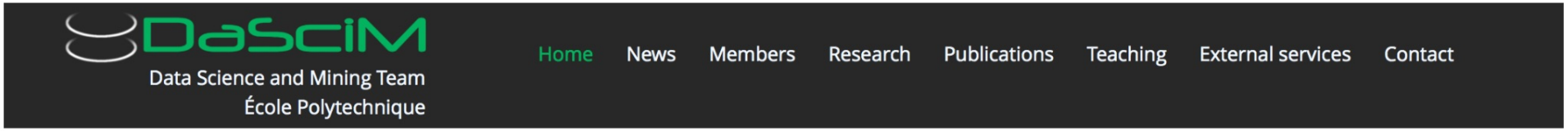
$$f(S) = \frac{1}{4}\left(m_s - E\left(m_s\right)\right)$$

Measures the difference between the number of edges in **S** and the expected number of edges **E(m_s)** in case of a configuration model

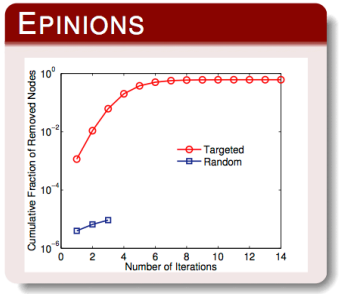
- Typically, a random graph model with the same degree sequence

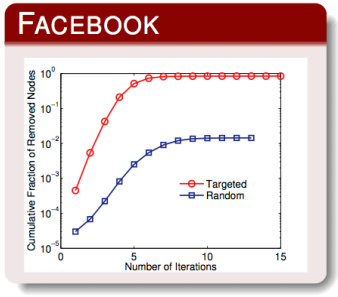


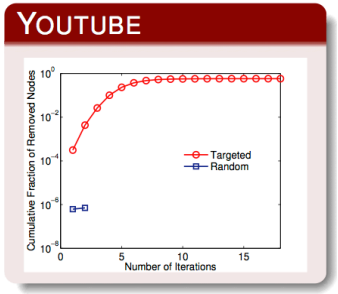
Graph Mining with degeneracy



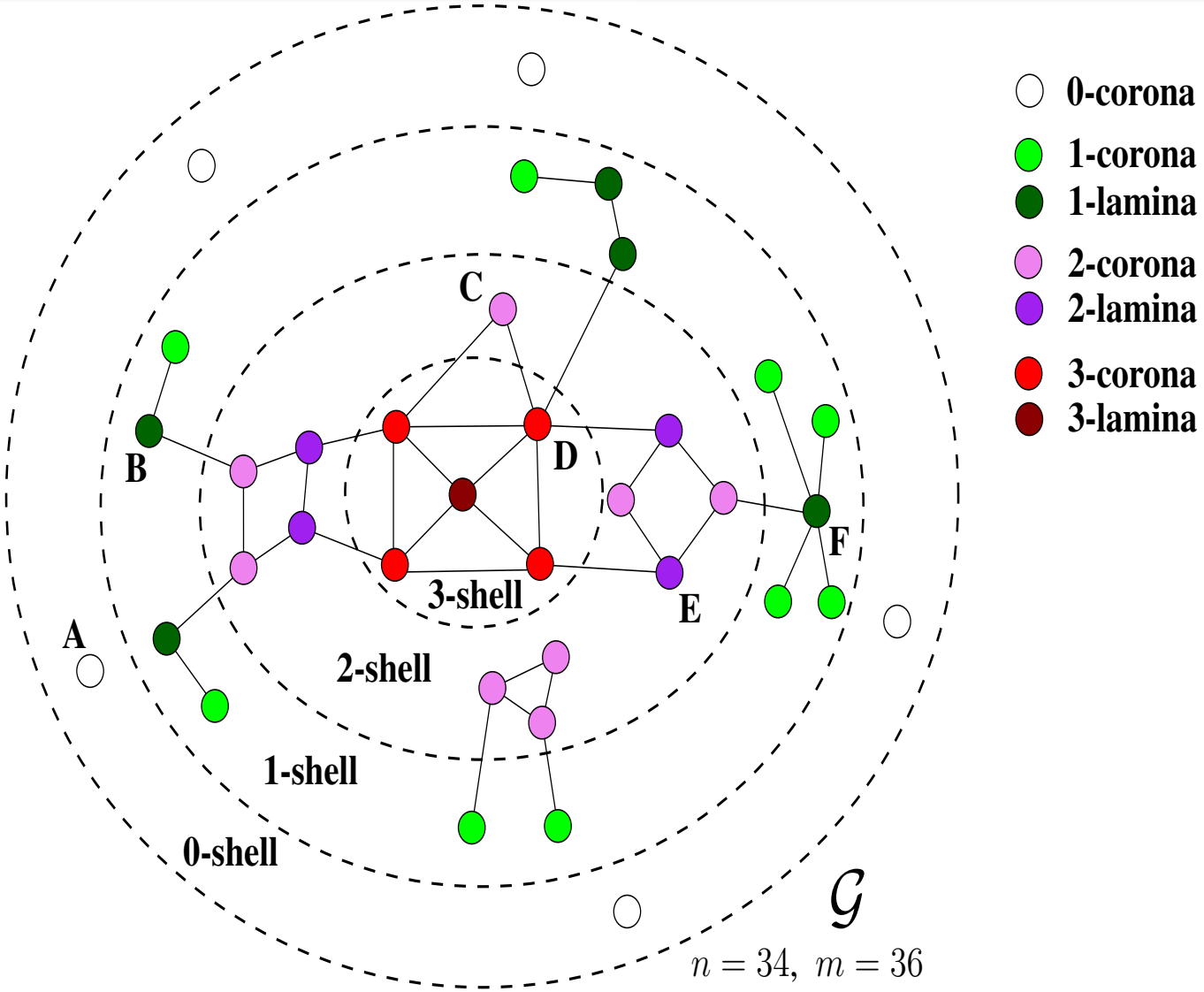
- Community detection & evaluation
  - Identifying groups of users highly collaborating among them





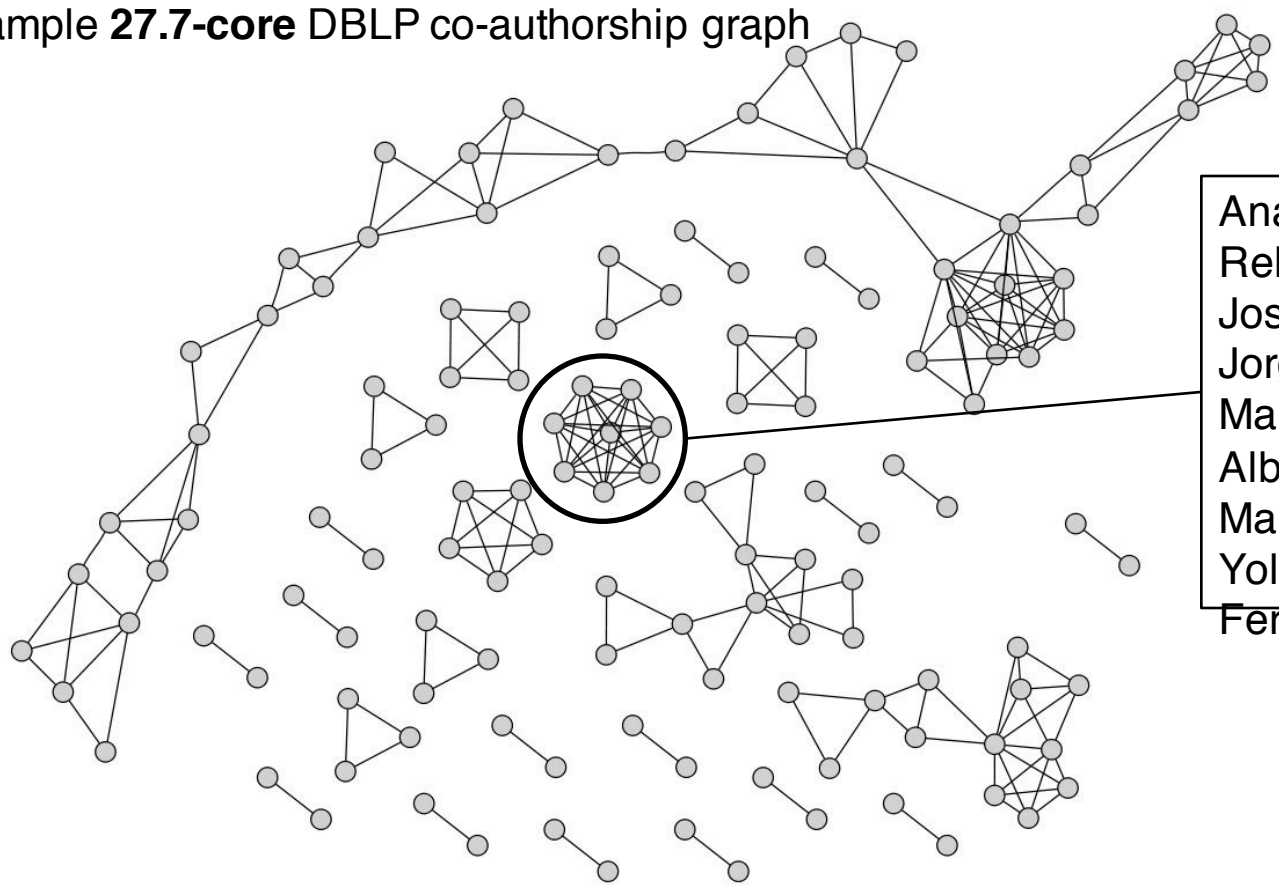


Graph Mining – k-core concept



Community detection and evaluation

Example **27.7-core** DBLP co-authorship graph

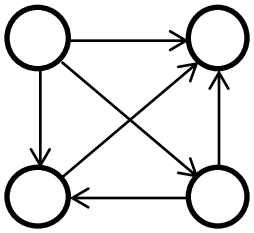


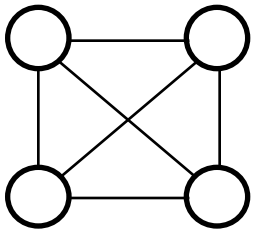
Ana Fernandez Vilas
Rebeca P. Diaz Redondo
Jose J. Pazos Arias
Jorge Garcia Duque
Martin Lopez Nores
Alberto Gil-Solla
Manuel Ramos Cabrer
Yolanda Blanco-Fernandez

http://www.graphdegeneracy.org/

Community detection and evaluation

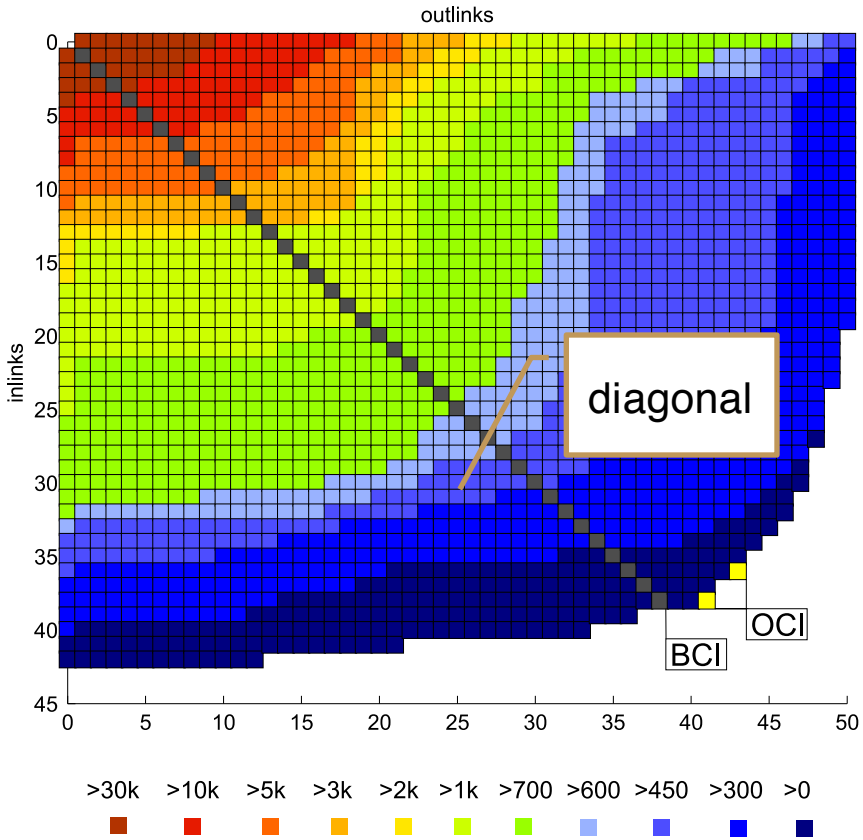
Degeneracy in directed graphs



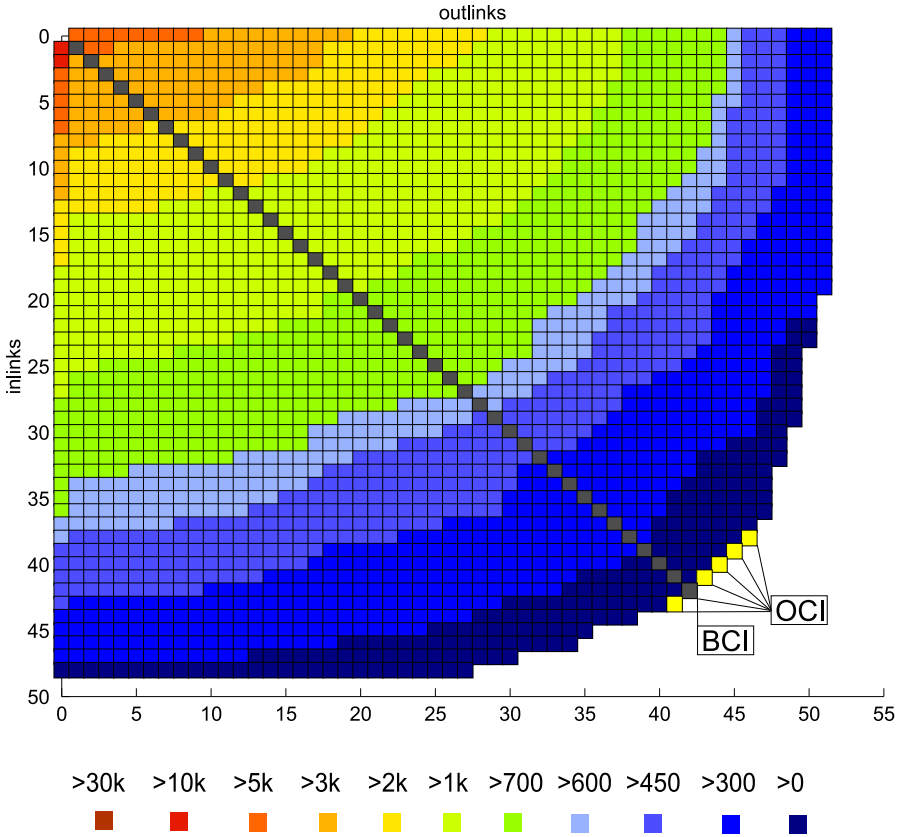


- Directed graphs:
  - WIKI - graph
  - DBLP & ARXIV – Citation graph
- Is there a degeneracy notion for directed graphs?
- We extend the k-core concept in directed graphs by applying a limit on in/out edges respectively
- Trade off between in/out edges can give us a more specific view of the cohesiveness and the “social" behavior

D-core matrix Wikipedia & DBLP



Wikipedia
The extreme D-core(38,41) contains 237 pages



DBLP
One of the extreme D-cores(38,46) contains 188 authors

The Extreme DBLP citation graph D-core

José A. Blakeley
Hector Garcia-Molina
Abraham Silberschatz
Umeshwar Dayal
Eric N. Hanson
Jennifer Widom
Klaus R. Dittrich
Nathan Goodman
Won Kim
Alfons Kemper
Guido Moerkotte
Clement T. Yu
M. Tamer Ã Zsu
Amit P. Sheth
Ming-Chien Shan
Richard T. Snodgrass
David Maier
Michael J. Carey
David J. DeWitt
Joel E. Richardson
Eugene J. Shekita
Waqar Hasan
Marie-Anne Neimat
Darrell Woelk
Roger King
Stanley B. Zdonik
Lawrence A. Rowe
Michael Stonebraker
Serge Abiteboul
Richard Hull
Victor Vianu
Jeffrey D. Ullman
Michael Kifer
Philip A. Bernstein
Vassos Hadzilacos
Elisa Bertino
Stefano Ceri
Georges Gardarin

Patrick Valduriez
Ramez Elmasri
Richard R. Muntz
David B. Lomet
Betty Salzberg
Shamkant B. Navathe
Arie Segev
Gio Wiederhold
Witold Litwin
Theo Härder
François Bancilhon
Raghu Ramakrishnan
Michael J. Franklin
Yannis E. Ioannidis
Henry F. Korth
S. Sudarshan
Patrick E. O'Neil
Dennis Shasha
Shamim A. Naqvi
Shalom Tsur
Christos H. Papadimitriou
Georg Lausen
Gerhard Weikum
Kotagiri Ramamohanarao
Maurizio Lenzerini
Domenico Saccà
Giuseppe Pelagatti
Paris C. Kanellakis
Jeffrey Scott Vitter
Letizia Tanca
Sophie Cluet
Timos K. Sellis
Alberto O. Mendelzon
Dennis McLeod
Calton Pu
C. Mohan
Malcolm P. Atkinson
Doron Rotem

Michel E. Adiba
Kyuseok Shim
Goetz Graefe
Jiawei Han
Edward Sciore
Rakesh Agrawal
Carlo Zaniolo
V. S. Subrahmanian
Claude Delobel
Christophe Lecluse
Michel Scholl
Peter C. Lockemann
Peter M. Schwarz
Laura M. Haas
Arnon Rosenthal
Erich J. Neuhold
Hans-Jorg Schek
Dirk Van Gucht
Hamid Pirahesh
Marc H. Scholl
Peter M. G. Apers
Allen Van Gelder
Tomasz Imielinski
Yehoshua Sagiv
Narain H. Gehani
H. V. Jagadish
Eric Simon
Peter Buneman
Dan Suciu
Christos Faloutsos
Donald D. Chamberlin
Setrag Khoshafian
Toby J. Teorey
Randy H. Katz
Miron Livny
Philip S. Yu
Stanley Y. W. Su
Henk M. Blanken

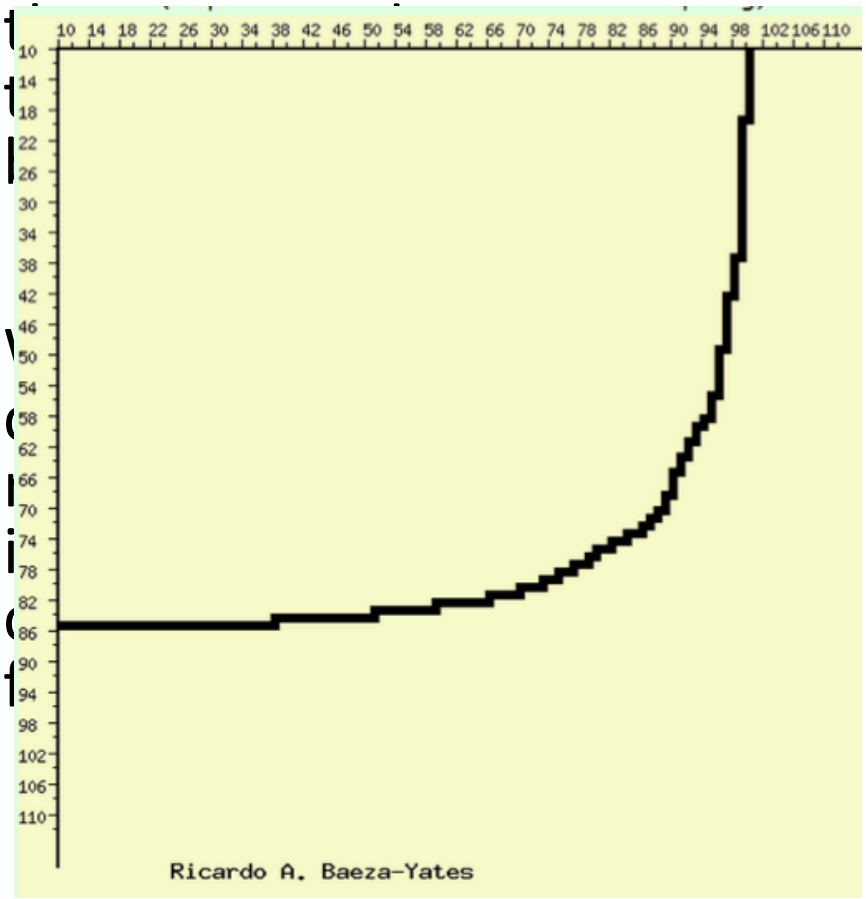
Peter Pistor
Matthias Jarke
Moshe Y. Vardi
Daniel BarbarÃ¡
Uwe Deppisch
H.-Bernhard Paul
Don S. Batory
Marco A. Casanova
Joachim W. Schmidt
Guy M. Lohman
Bruce G. Lindsay
Paul F. Wilms
Z. Meral Özsoyoglu
Gultekin Özsoyoglu
Kyu-Young Whang
Shahram Ghandeharizadeh
Tova Milo
Alon Y. Levy
Georg Gottlob
Johann Christoph Freytag
Klaus Küspert
Louiqa Raschid
John Mylopoulos
Alexander Borgida
Anand Rajaraman
Joseph M. Hellerstein
Masaru Kitsuregawa
Sumit Ganguly
Rudolf Bayer
Raymond T. Ng
Daniela Florescu
Per-Åke Larson
Hongjun Lu
Ravi Krishnamurthy
Arthur M. Keller
Catriel Beeri
Inderpal Singh Mumick
Oded Shmueli

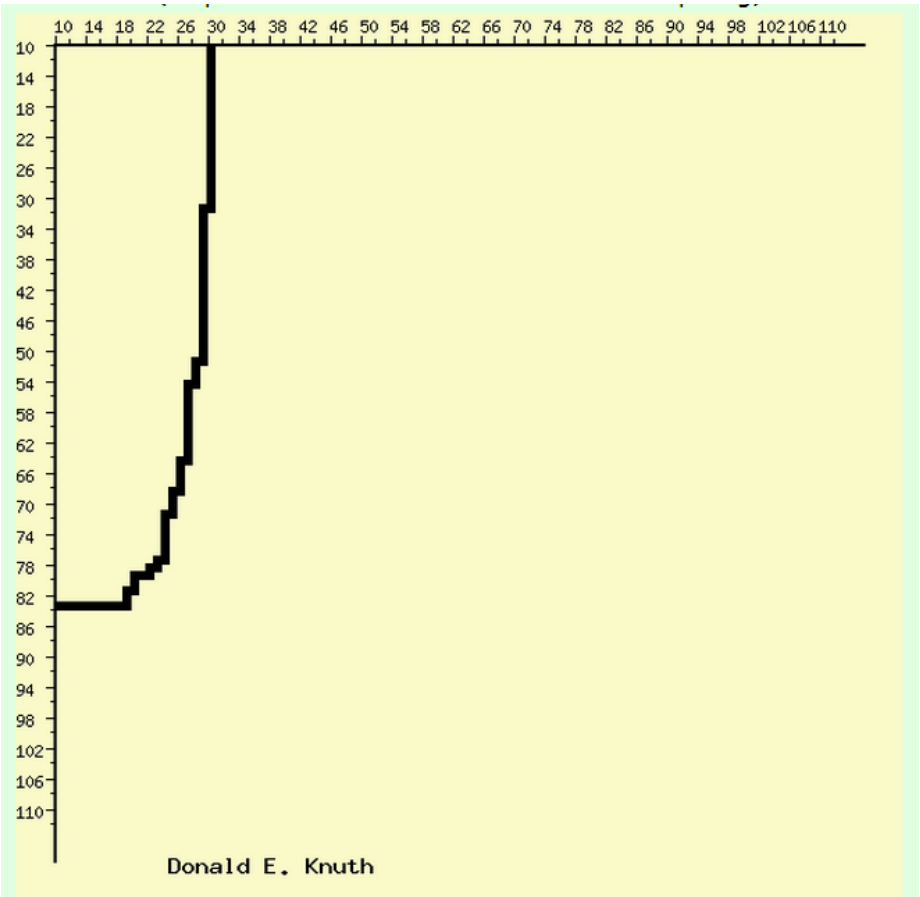
George P. Copeland
Peter Dadam
Susan B. Davidson
Donald Kossmann
Christophe de Maindreville
Yannis Papakonstantinou
Kenneth C. Sevcik
Gabriel M. Kuper
Peter J. Haas
Jeffrey F. Naughton
Nick Roussopoulos
Bernhard Seeger
Georg Walch
R. Erbe
Balakrishna R. Iyer
Ashish Gupta
Praveen Seshadri
Walter Chang
Surajit Chaudhuri
Divesh Srivastava
Kenneth A. Ross
Arun N. Swami
Donovan A. Schneider
S. Seshadri
Edward L. Wimmers
Kenneth Salem
Scott L. Vandenberg
Dallan Quass
Michael V. Mannino
John McPherson
Shaul Dar
Sheldon J. Finkelstein
Leonard D. Shapiro
Anant Jhingran
George Lapis

D-Core frontier for individuals

- The frontier of an individual: defined by

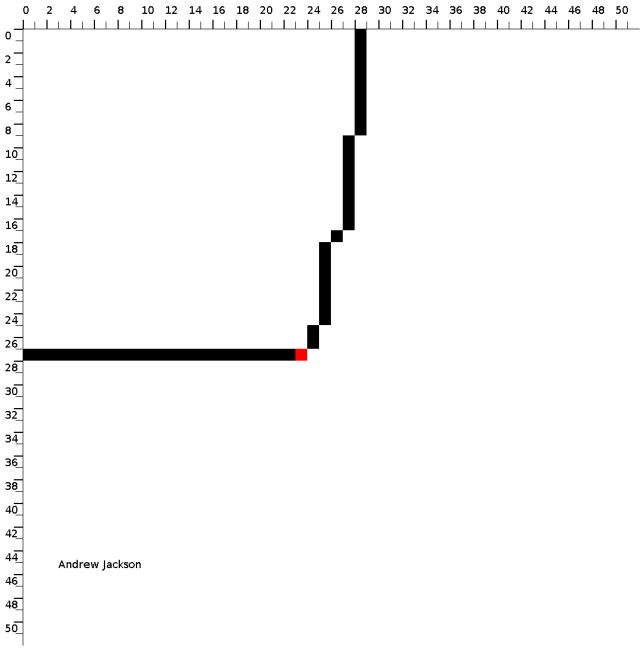
outlinks



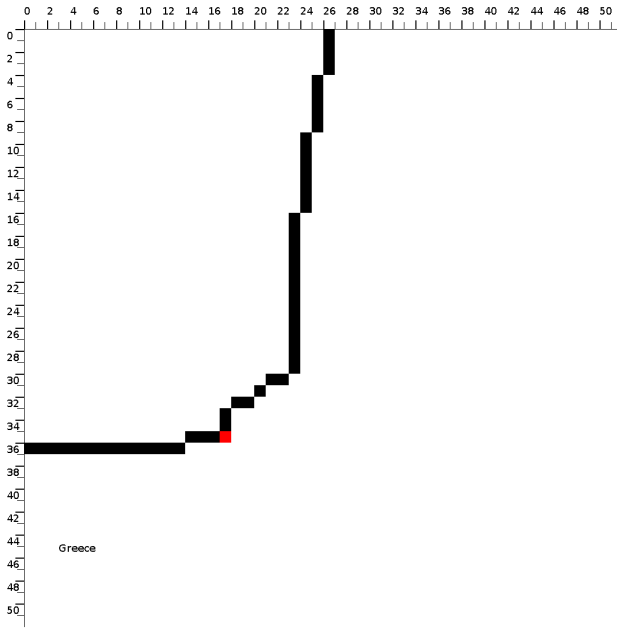


http://www.graphdegeneracy.org/

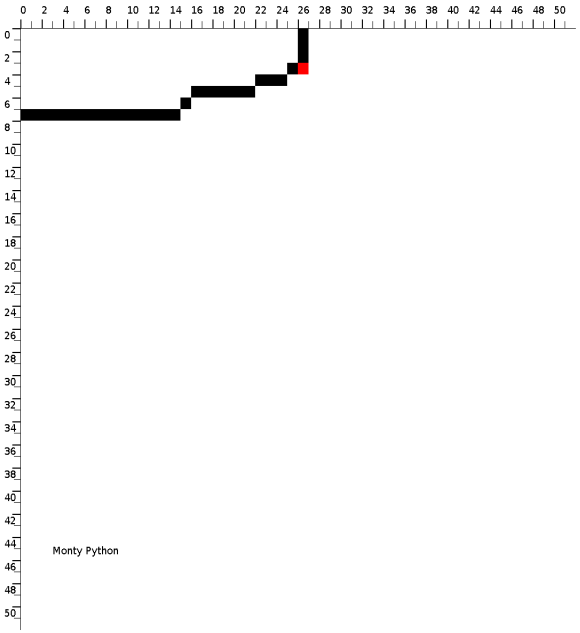
Thematic D-core frontiers - Wikipedia



“Andrew Jackson”

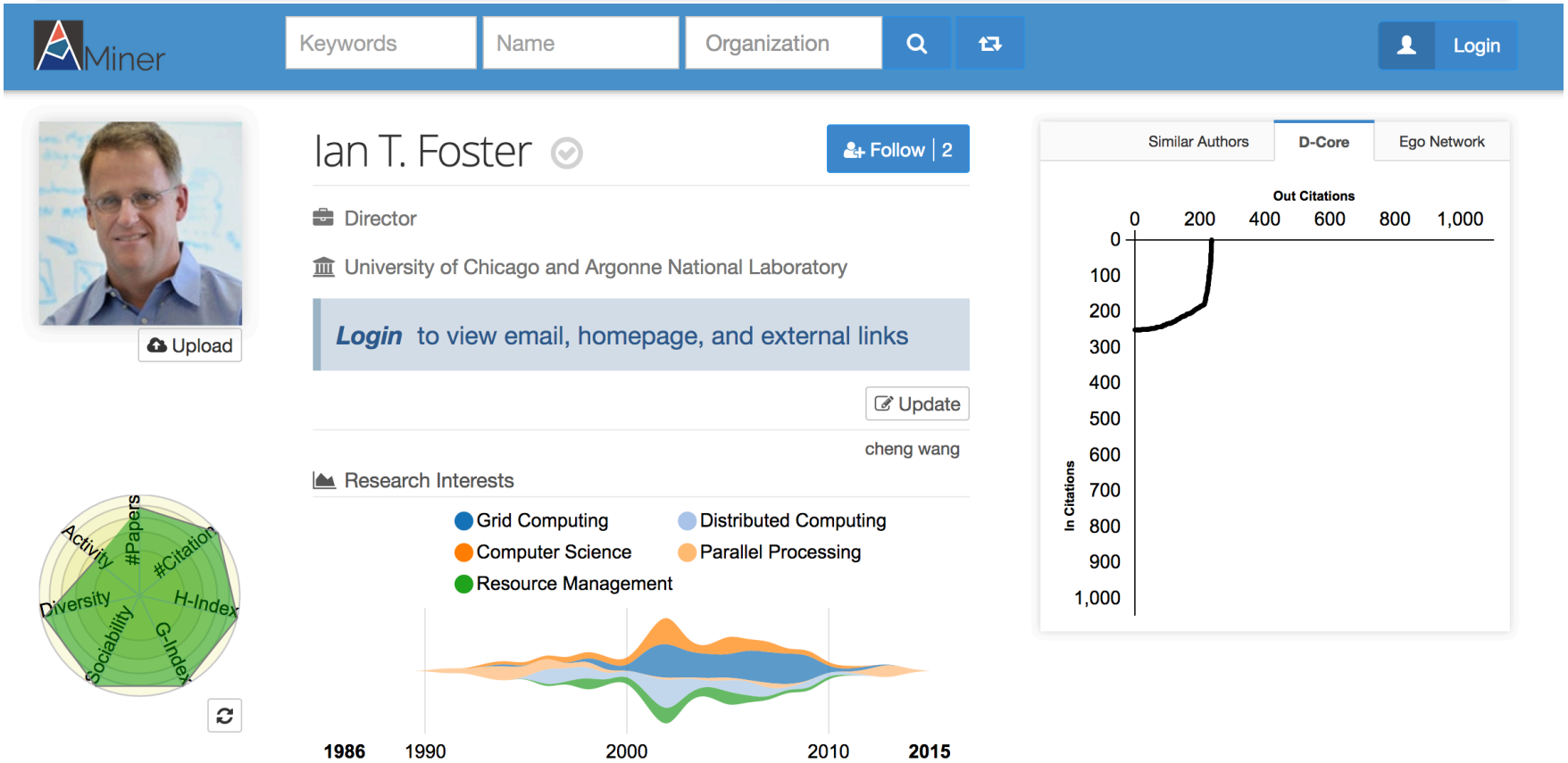


“Greece”



“Monty Python”

D-core adopted by aminer.org



https://cn.aminer.org/profile/ian-t-foster/53f48850dabfaee4dc8b2045

Outline

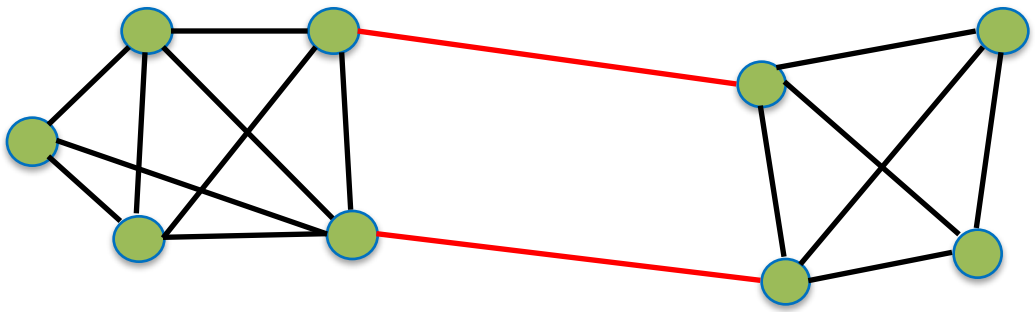
- Motivating Graph Mining and basic notions
- Graph Exploration and Generation
  - Degree distribution – power laws
  - Graph generators
- Machine Learning for Graphs
  - **Community detection - clustering**
  - Supervised learning with Graph Kernels

Notations

- Given Graph G=(V,E) undirected:
  - Vertex Set V={v_1,.....v_n}, Edge e_{ij} between v_i and v_j
    - we assume weight w_{ij}>0 for e_{ij}
  - |V| : number of vertices
  - d_i degree of v_i : $d_i = \sum_{v_j \in V} w_{ij}$
  - $v(V) = \sum_{v_i \in V} d_i$
  - for $A \subset V$ $\bar{A} = V - A$
  - Given $\mathrm{A}, \mathrm{B} \subset V$ & $\mathrm{A} \cap B = \emptyset, w(A, B) = \sum_{v_i \in A, vj \in B} w_{ij}$
  - D : Diagonal matrix where $\mathrm{D}(\mathrm{i}, \mathrm{i}) = d_i$
  - W : Adjacency matrix $\mathrm{W}(\mathrm{i}, j) = w_{ij}$

Graph-Cut

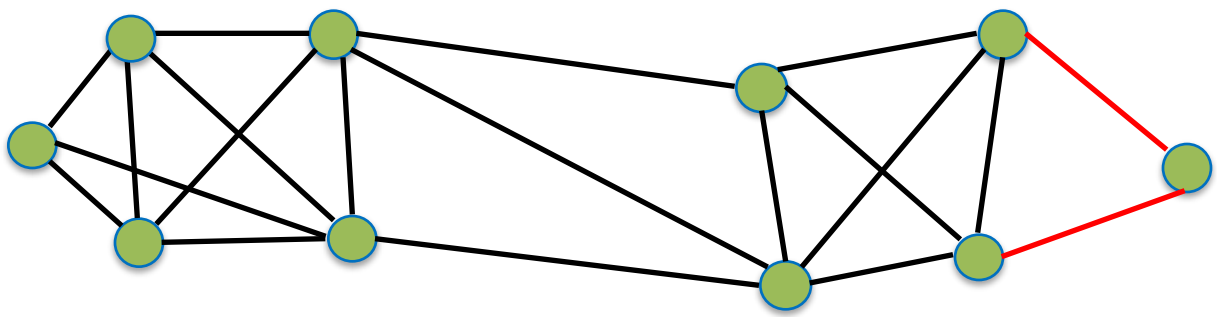
- For k clusters:
  - $cut(A_1,..,Ak) = 1/2 \sum_{i=1}^{k} w(Ai, \overline{A_i})$
    - undirected graph:1/2 we count twice each edge



- Min-cut:Minimize the edges' weight a cluster shares with the rest of the graph

Min-Cut

- Easy for k=2 : Mincut(A_1,A_2)
  - Stoer and Wagner: “A Simple Min-Cut Algorithm”
- In practice one vertex is separated from the rest
  - The algorithm is drawn to outliers



Normalized Graph Cuts

- We can normalize by the size of the cluster (size of sub-graph) :
  - number of Vertices (Hagen and Kahng, 1992):

    $$Ratiocut(A_1, \dots Ak) = \sum_{i=1}^{k} \frac{cut(Ai, \overline{A_i})}{|Ai|}$$

  - sum of weights (Shi and Malik, 2000) :

    $$Ncut(A_1, \dots Ak) = \sum_{i=1}^{k} \frac{cut(Ai, \overline{A_i})}{v(A_i)}$$

- Optimizing these functions is NP-hard
- Spectral Clustering provides solution to a relaxed version of the above

From Graph Cuts to Spectral Clustering

- For simplicity assume k=2:
  - Define $f: V \rightarrow \mathbb{R}$ for Graph G :

$$f_i = \begin{cases} 1 & v_i \in A \\ -1 & v_i \in \overline{A} \end{cases}$$

- Optimizing the original cut is equivalent to an optimization of:

$$\sum_{i,j=1}^{n} w_{ij}\left(f_i - f_j\right)^2$$

$$= \sum_{v_i \in A, vj \in \overline{A}} w_{ij}(1+1)^2 + \sum_{v_i \in \overline{A}, vj \in A} w_{ij}(-1-1)^2$$

$$= \mathbf{8} * \boldsymbol{cut}\left(\boldsymbol{A}, \overline{\boldsymbol{A}}\right)$$

Graph Laplacian

- How is the previous useful in Spectral clustering?

$$\sum_{i,j=1}^{n} w_{ij}(f_i - fj)^2$$

$$= \sum_{i,j=1}^{n} w_{ij} {f_i}^2 - 2\sum_{i,j=1}^{n} w_{ij} f_i f_j + \sum_{i,j=1}^{n} w_{ij} {f_j}^2$$

$$= \sum_{i,j=1}^{n} d_i {f_i}^2 - 2\sum_{i,j=1}^{n} w_{ij} f_i f_j + \sum_{i,j=1}^{n} d_j {f_j}^2$$

$$= 2\left(\sum_{i,j=1}^{n} d_{ii} {f_i}^2 - \sum_{i,j=1}^{n} w_{ij} f_i f_j\right)$$

$$= 2\left(\boldsymbol{f}^T \boldsymbol{D} \boldsymbol{f} - \boldsymbol{f}^T \boldsymbol{W} \boldsymbol{f}\right) = 2\boldsymbol{f}^T(\boldsymbol{D} - \boldsymbol{W})\boldsymbol{f} = 2\boldsymbol{f}^T \boldsymbol{L} \boldsymbol{f}$$

- **f:**a single vector with the cluster assignments of the vertices
- **L=D-W :** the Laplacian of a graph

Properties of L

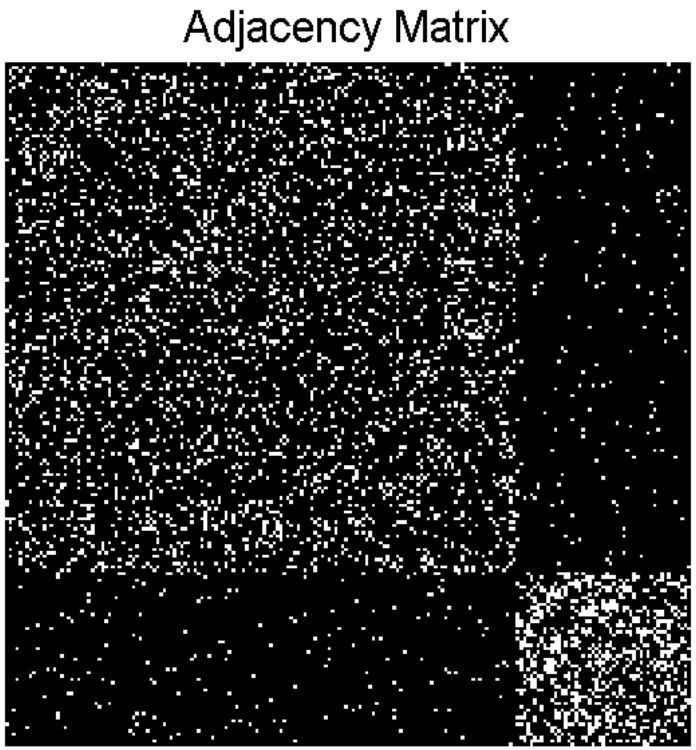
- L is
  - Symmetric
  - Positive
  - Semi-definite
- The smallest eigenvalue of L is 0
  - The corresponding eigenvector is $\mathbb{1}$
- L has n non-negative, real valued eigenvalues
  - $0 = \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$

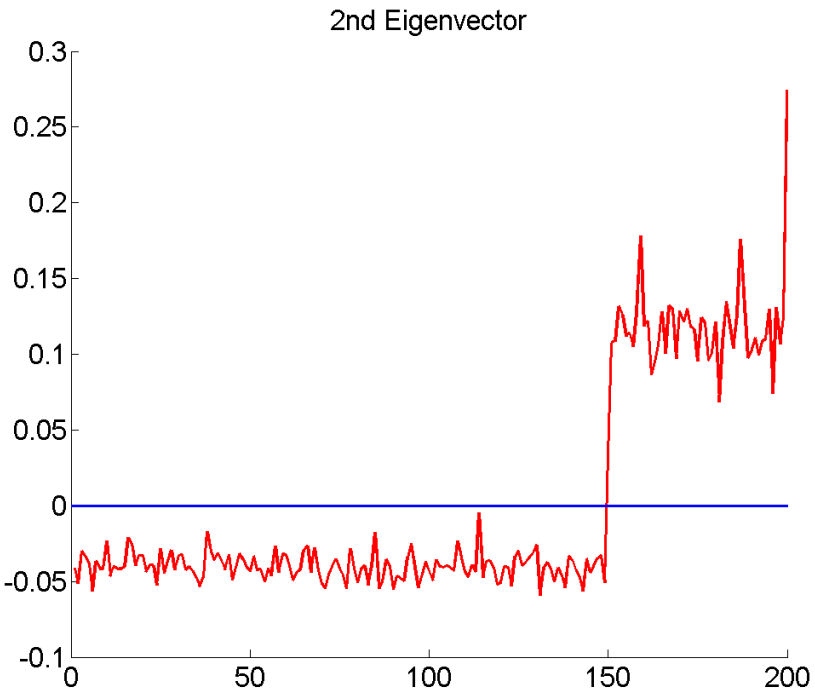
Two Way Cut from the Laplacian

- We could solve $min_f f^T L f$ where $f \in \{-1,1\}^n$
- NP-Hard for discrete cluster assignments
  - Relax the constraint $to\ f \in R^n$:

    $min_f f^T L f$ subject to $f^T f = n$
- The solution to this problem is given by:
  - (**Rayleigh-Ritz Theorem**) the eigenvector corresponding to smallest eigenvalue: 0 and the corresponding eigenvector (full of 1s) offers no information
- We use the second eigenvector as an approximation
  - $f_i > 0$ the vertex belongs to one cluster , $f_i < 0$ to the other

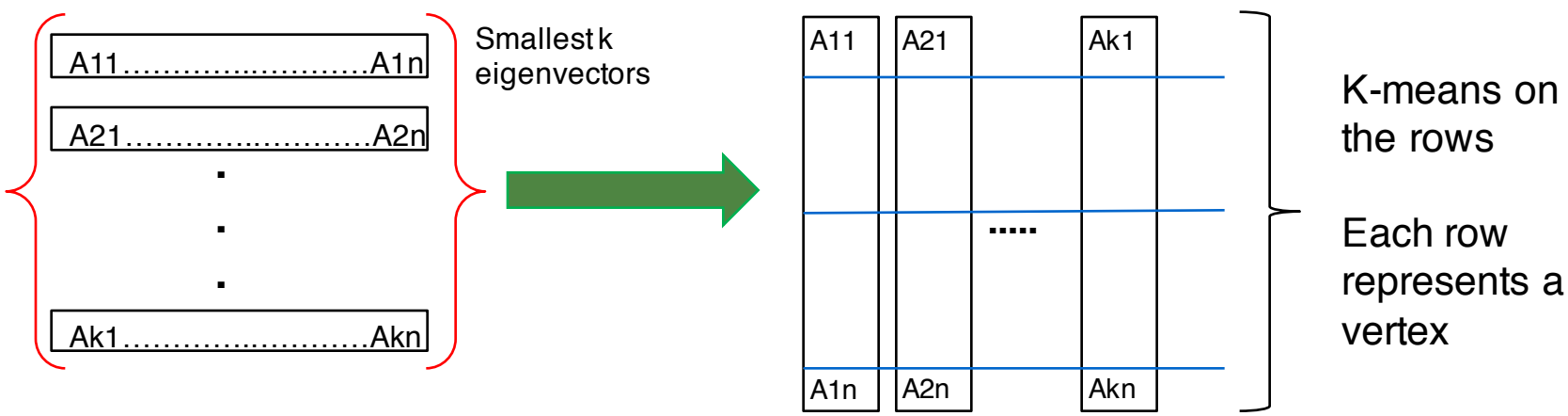
Example


Adjacency Matrix


2nd Eigenvector
0.3
0.25
0.2
0.15
0.1
0.05
0
-0.05
-0.1
0
50
100
150
200

Multi-Way Graph Partition

- The cluster assignment is given by the smallest k eigenvectors of L
- The real values need to be converted to cluster assignments
  - We use k-means to cluster the rows
  - We can substitute L with L_{sym}



References – Graph clustering

- UIrike von Luxburg, A Tutorial on Spectral Clustering, Statistics and Computing, 2007
- Davis, C., W. M. Kahan (March 1970). The rotation of eigenvectors by a perturbation. III. SIAM J. Numerical Analysis 7
- Shi, Jianbo, and Jitendra Malik. "Normalized cuts and image segmentation, "*Pattern Analysis and Machine Intelligence, IEEE Transactions on* (2000).
- Mechthild Stoer and Frank Wagner. 1997. A simple min-cut algorithm. *J. ACM*
- Ng, Jordan & Weiss, K-means algorithm on the embedded eigen-space, NIPS 2001
- Hagen, L. Kahng, , "New spectral methods for ratio cut partitioning and clustering," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on* , 1992

References (modularity)

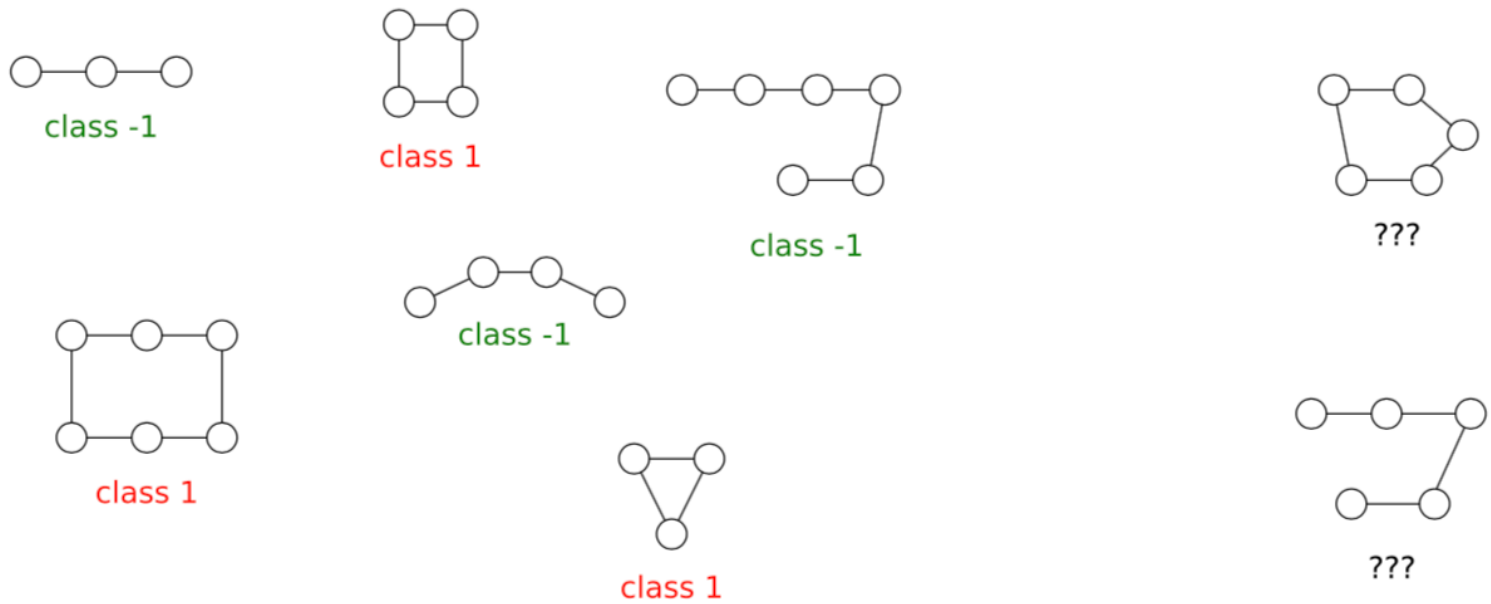
- M.E.J. Newman and M. Girvan. Finding and evaluating community structure in networks. Physical Review E 69(02), 2004.
- M.E.J. Newman. Modularity and community structure in networks. PNAS, 103(23), 2006.
- S.E. Schaeffer. Graph clustering. Computer Science Review 1(1), 2007.
- S. Fortunato. Community detection in graphs. Physics Reports 486 (3-5), 2010.
- M. Coscia, F. Giannotti, and D. Pedreschi. A classification for community discovery methods in complex networks. Statistical Analysis and Data Mining 4 (5), 2011.
- A. Arenas, J. Duch, A. Fernandez, and S. Gomez. Size reduction of complex networks preserving modularity. New J. Phys., 9(176), 2007.
- M. Girvan and M.E.J. Newman. Community structure in social and biological networks. PNAS 99(12), 2002.
- U. Brandes, D. Delling, M. Gaertler, R. Gorke, M. Hoefer, Z. Nikoloski, and D. Wagner. On Modularity Clustering. IEEE TKDE 20(2), 2008.
- M.E.J. Newman. Fast algorithm for detecting community structure in networks. Phys. Rev. E 69, 2004.
- A. Clauset, M.E.J. Newman, and C. Moore. Finding community structure in very large networks. Phys. Rev. E 70, 2004.

Outline

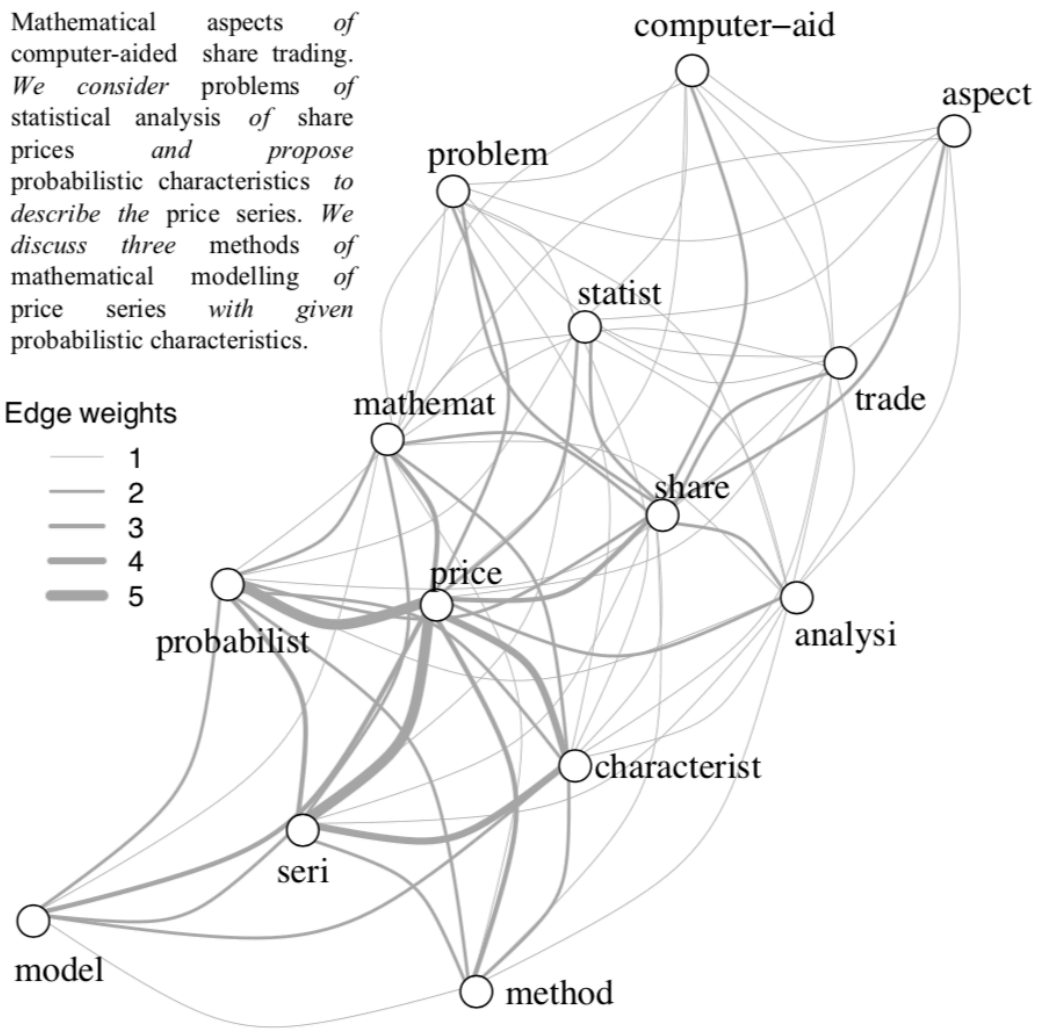
- Motivating Graph Mining and basic notions
- Graph Exploration and Generation
  - Degree distribution – power laws
  - Graph generators
- Machine Learning for Graphs
  - Community detection
  - **Supervised learning with Graph Kernels**

Machine Learning for Graphs

- Node classification
- Graph clustering
- Link Prediction:
- Graph classification



Motivation – Text categorization



Given a text, create a graph where

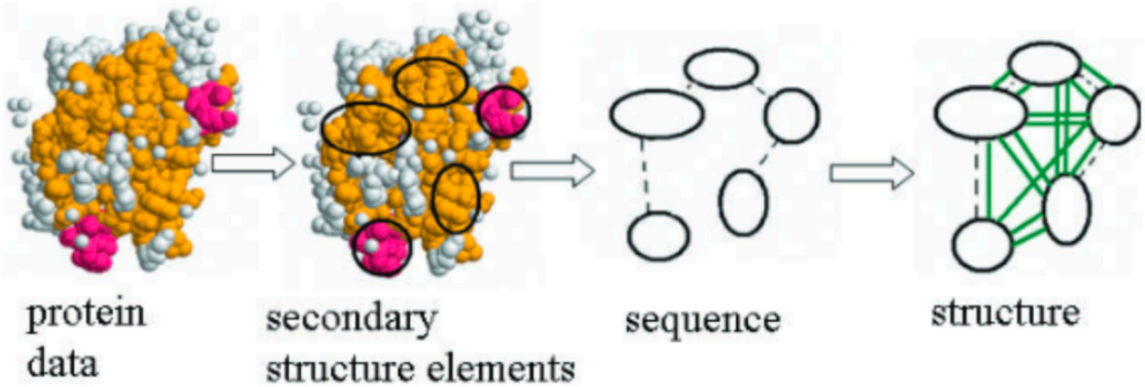
- vertices correpond to terms
- two terms are linked to each other if they co-occur within a fixed-size sliding window

Rousseau et al. "Text categorization as a graph classification problem.". In ACL'15

Motivation – Protein Function Prediction

For each protein, create a graph that contains information about its

- structure
- sequence
- chemical properties



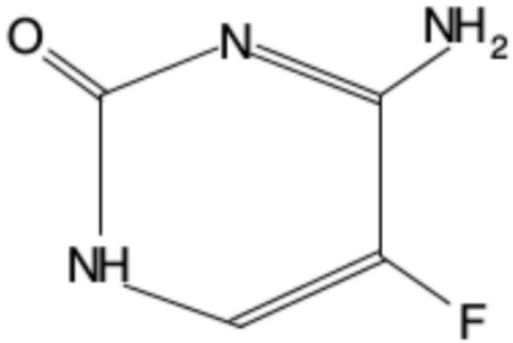
Use graph kernels to

- measure structural similarity between proteins
- predict the function of proteins

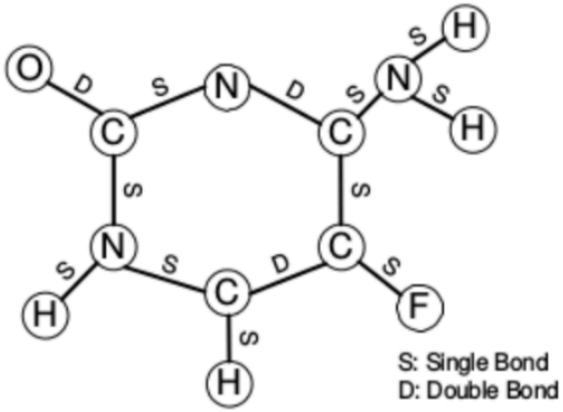
Borgwardt et al. "Protein function prediction via graph kernels". Bioinformatics 21

Motivation – Chemical compound classification

Represent each chemical compound as a graph



→



Use a frequent subgraph discovery algorithm to discover the substructures that occur above a certain support constraint

Perform feature selection

Use the remaining substructures as features for classification

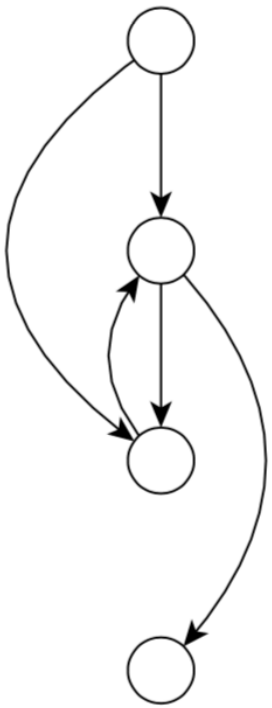
Deshpande et al. "Frequent substructure-based approaches for classifying chemical compounds". TKDE 17(8)

Motivation – Malware detection

Given a computer program, create its control flow graph

```
        processed_pages.append(processed_page)
        visited += 1
        links = extract_links(html_code)
        for link in links:
            if link not in visited_links:
                links_to_visit.append(link)

    return create_vocabulary(processed_pages)


def parse_page(html_code):
    punct = re.compile(r'([^A-Za-z0-9])')
    soup = BeautifulSoup(html_code, 'html.parser')
    text = soup.get_text()
    processed_text = punct.sub(" ", text)
    tokens = processed_text.split()
    tokens = [token.lower() for token in tokens]
    return tokens


def create_vocabulary(processed_pages):
    vocabulary = {}
    for processed_page in processed_pages:
        for token in processed_page:
            if token in vocabulary:
                vocabulary[token] += 1
            else:
                vocabulary[token] = 1

    return vocabulary
```

$\rightarrow$



Compare the control flow graph of the problem against the set of control flow graphs of known malware

If it contains a subgraph isomporphic to these graphs $\rightarrow$ malicious code inside the program

Gascon et al. "Structural detection of android malware using embedded call graphs". In AISec'13

Graph similarity

Graph classification very related to graph comparison

Example

$f($  $)$

$+$

$k{-}nn$

$=$ graph classification

Although graph comparison seems a tractable problem, it is very **complex**

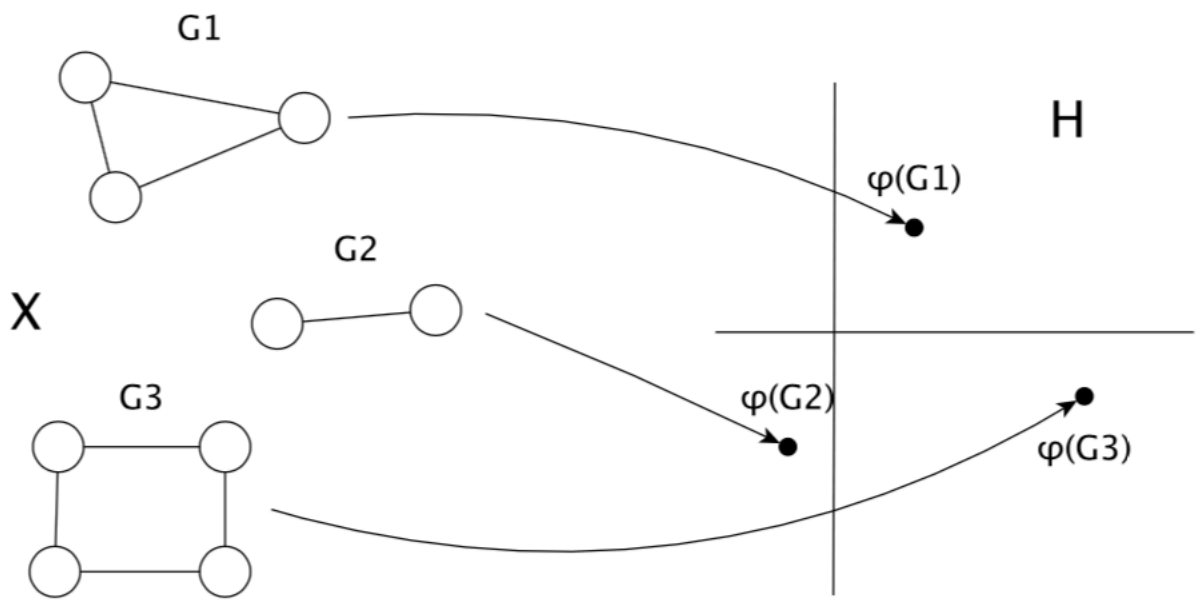
We are interested in algorithms capable of measuring the similarity between two graphs in **polynomial** time

Graph Kernels

Definition (Graph Kernel)

A graph kernel $k : \mathcal{G} \times \mathcal{G} \to \mathcal{R}$ is a kernel function over a set of graphs $\mathcal{G}$

- It is equivalent to an inner product of the embeddings $\phi : \mathcal{X} \to \mathbb{H}$ of a pair of graphs into a Hilbert space: $k(G_1, G_2) = \langle \phi(G_1), \phi(G_2) \rangle$
- Makes the whole family of kernel methods (e.g. SVMs) applicable to graphs



Graph invariants

We saw that proving that two graphs are isomorphic is not a simple task

It is much simpler to show that two graphs are not isomorphic by finding a property that only one of the two graphs has. Such a property is called a *graph invariant*

Definition (Graph Invariant)

A graph invariant is a numerical property of graphs for which any two isomorphic graphs must have the same value

Some examples of graph invariants include:

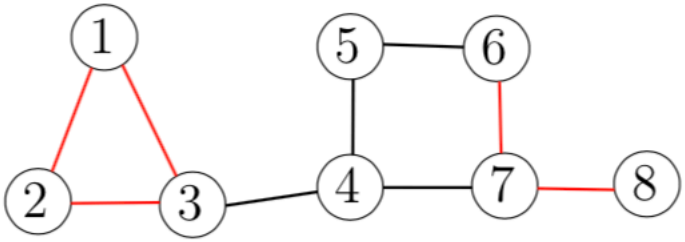
1. number of vertices
2. number of edges
3. number of spanning trees
4. degree sequence
5. spectrum

Substructures for similarity

A large number of graph kernels compare substructures of graphs that are computable in polynomial time:

- walks
- shortest path lengths
- cyclic patterns
- rooted subtrees
- graphlets

⋮



Shervashidze et al. "Efficient graphlet kernels for large graph comparison.". In AISTATS'09

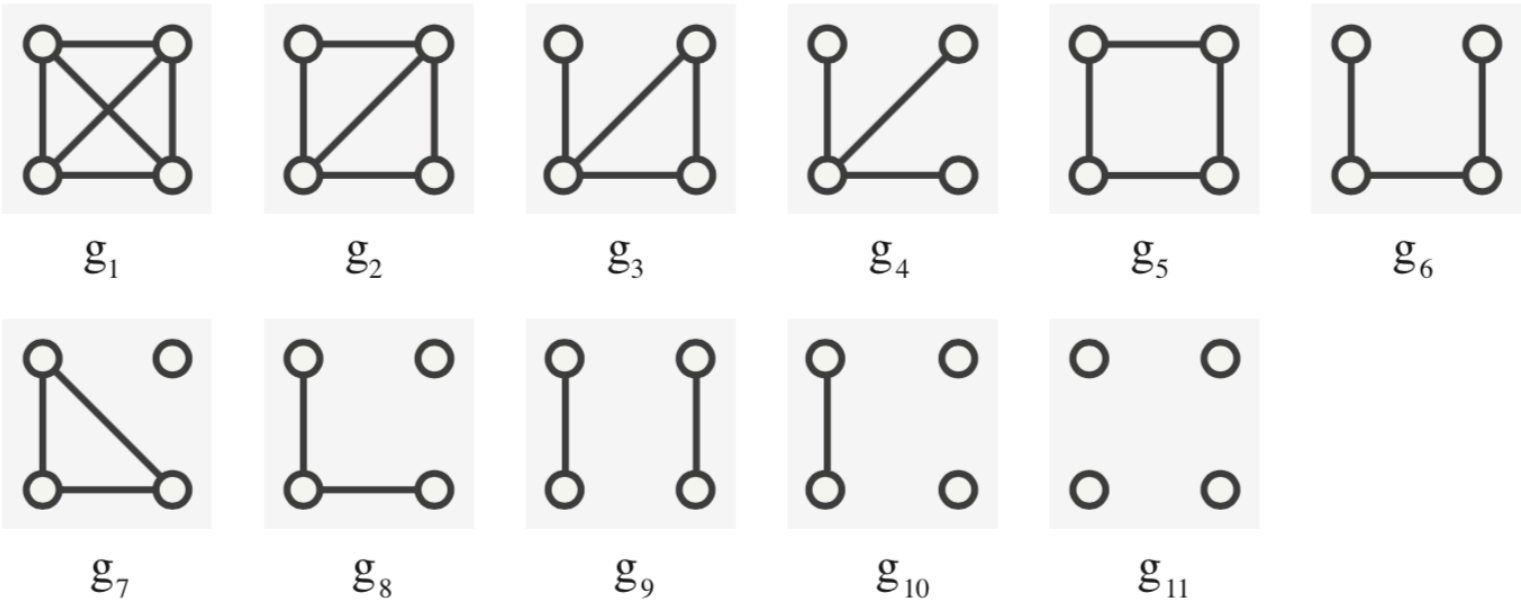
Graphlet Kernel

The graphlet kernel compares graphs by counting *graphlets*

A graphlet corresponds to a small subgraph

- typically of 3,4 or 5 vertices

Below is the set of graphlets of size 4



Shervashidze et al. "Efficient graphlet kernels for large graph comparison". In AISTATS'09

Graphlet Kernel

Let $\mathcal{G} = \{graphlet_1, graphlet_2, \ldots, graphlet_r\}$ be the set of size-k graphlets

Let also $f_G \in \mathcal{N}^r$ be a vector such that its i-th entry is $f_{G,i} = \#(graphlet_i \sqsubseteq G)$

The graphlet kernel is defined as:

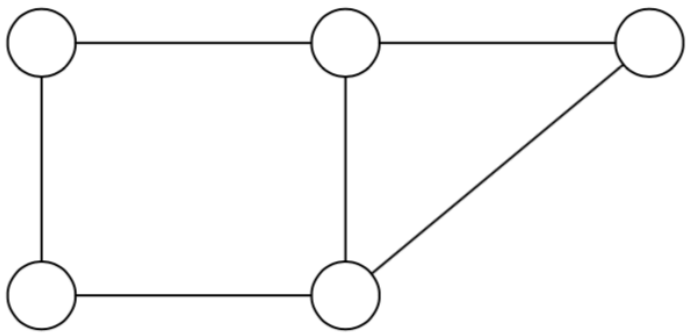
$$k(G_1, G_2) = f_{G_1}^\top f_{G_2}$$

Problems:

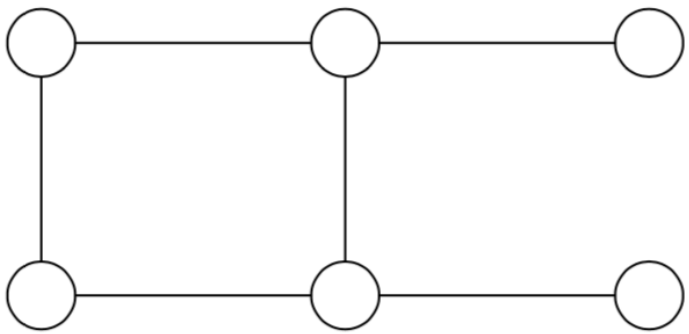
- There are $\binom{n}{k}$ size-k subgraphs in a graph
- Exaustive enumeration of graphlets is very expensive

  Requires $O(n^k)$ time
- For labeled graphs, the number of graphlets increases further

Graphlet Kernel



G_1



G_2

The vector representations of the graphs above according to the set of graphlets of size 4 is:

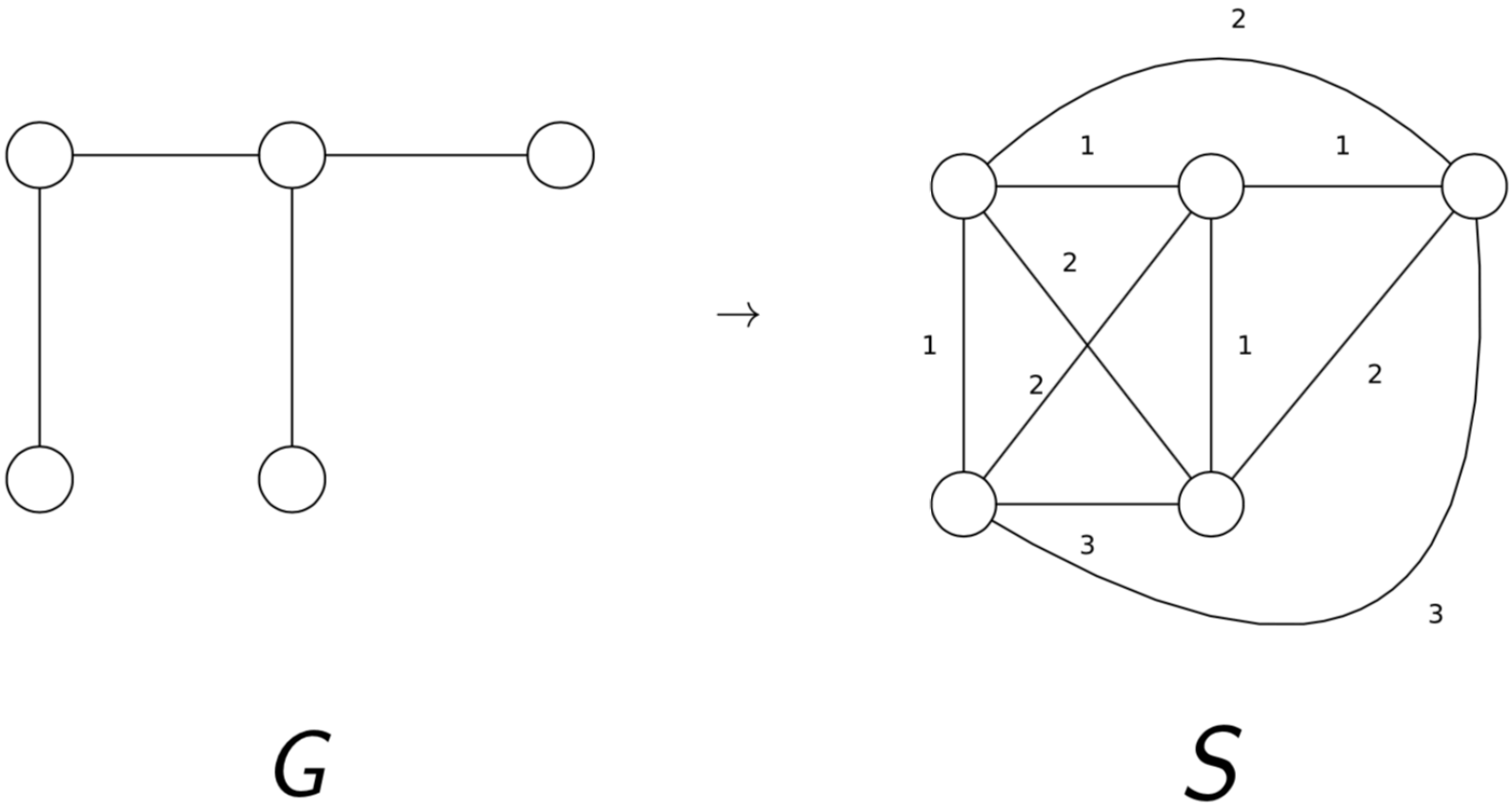
$$f_{G_1} = (0, 0, 2, 0, 1, 2, 0, 0, 0, 0, 0)^T$$
$$f_{G_2} = (0, 0, 0, 2, 1, 5, 0, 4, 0, 3, 0)^T$$

Hence, the value of the kernel is:

$$k(G_1, G_2) = f_{G_1}^\top f_{G_2} = 11$$

Shortest Path Kernel

Floyd-transformation



Shortest Path Kernel

Given the Floyd-transformed graphs $S_1 = (V_1, E_1)$ and $S_2 = (V_2, E_2)$ of G_1 and G_2, the shortest path kernel is defined as:

$$k(G_1, G_2) = \sum_{e_1 \in E_1} \sum_{e_2 \in E_2} k^{(1)}_{walk}(e_1, e_2)$$

where $k^{(1)}_{walk}$ is a kernel on edge walks of length 1

- For unlabeled graphs, it can be:

$$k^{(1)}_{walk}(e_1, e_2) = \delta(\ell(e_1), \ell(e_2)) = \begin{cases} 1 & \text{if } \ell(e_1) = \ell(e_2), \\ 0 & \text{otherwise} \end{cases}$$

  where $\ell(e)$ gives the label of edge e

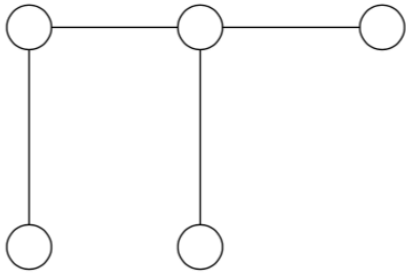
- For labeled graphs, it can be:

$$k^{(1)}_{walk}(e_1, e_2) = \begin{cases} 1 & \text{if } \ell(e_1) = \ell(e_2) \wedge \ell(e_1^1) = \ell(e_2^1) \wedge \ell(e_1^2) = \ell(e_2^2), \\ 0 & \text{otherwise} \end{cases}$$

  where e^1, e^2 are the two endpoints of e

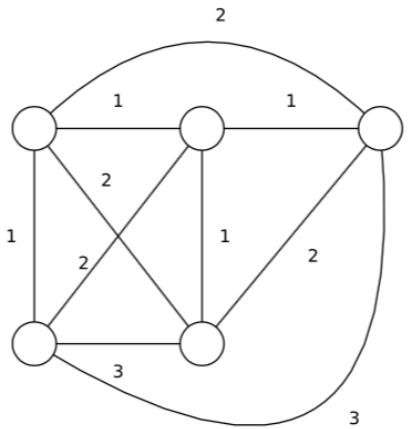
Shortest Path Kernel

Floyd-transformations

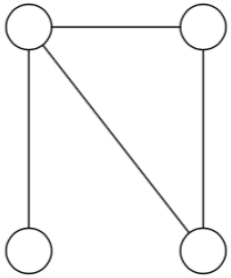


G_1

$\rightarrow$

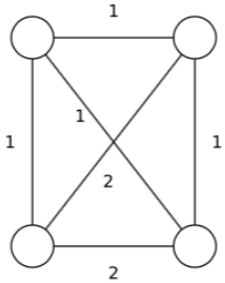


S_1



G_2

$\rightarrow$



S_2

Shortest Path Kernel

In S_1 we have:

- 4 edges with label 1
- 4 edges with label 2
- 2 edges with label 3

In S_2 we have:

- 4 edges with label 1
- 2 edges with label 2

Hence, the value of the kernel is:

$$k(G_1, G_2) = \sum_{e_1 \in E_1} \sum_{e_2 \in E_2} k^{(1)}_{walk}(e_1, e_2) = 4 * 4 + 4 * 2 = 24$$

Shortest Path Kernel

Computing the shortest path kernel includes:

- Computing shortest paths for all pairs of vertices in the two graphs: $\mathcal{O}(n^3)$
- Comparing all pairs of shortest paths from the two graphs: $\mathcal{O}(n^4)$

Hence, runtime is $\mathcal{O}(n^4)$

Problems:

- Very high complexity for large graphs
- Shortest-path graphs may lead to memory problems on large graphs

Relevant publications

- C. Giatsidis, D. Thilikos, and M. Vazirgiannis, D-cores: Measuring Collaboration of Directed Graphs Based on Degeneracy. Knowledge and Information Systems Journal, Springer, 2012.
- C. Giatsidis, K. Berberich, D. M. Thilikos, M. Vazirgiannis, Visual exploration of collaboration networks based ongraph degeneracy, *ACM KDD*, 2012.
- C. Giatsidis, D. Thilikos, and M. Vazirgiannis, D-cores: Measuring Collaboration of Directed Graphs Based on Degeneracy. *IEEE ICDM*, 2011,
- C. Giatsidis, D. Thilikos, and M. Vazirgiannis, Evaluating Cooperation in Communities with the k-Core Structure. *ACM/IEEE ASONAM*, 2011.
- F. D. Malliaros and M. Vazirgiannis, To Stay or Not to Stay: Modeling Engagement Dynamics in Social Graphs. *ACM CIKM*, 2013.
- F. D. Malliaros and M. Vazirgiannis, Clustering and Community Detection in Directed Networks: A Survey. *Physics Reports*, 533(4), Elsevier, 2013.
- F. D. Malliaros and M. Vazirgiannis, Vulnerability Assessment in Social Networks under Cascade-based Node Departures, *EPL (Europhysics Letters)*, 11(6), 2015.
- C. Giatsidis, F.D. Malliaros, D. Thilikos, and M. Vazirgiannis, CORECLUSTER: A Degeneracy Based Graph Clustering Framework. *AAAI*, 2014.
- F.D. Malliaros, V. Megalooikonomou and C. Faloutsos. Estimating Robustness in Large Social Graphs. *Knowledge and Information Systems (KAIS)*, Springer, 2015.
- M.-E. G. Rossi, F.D. Malliaros, and M. Vazirgiannis, Spread It Good, Spread It Fast: Identification of Influential Nodes in Social Networks. *WWW*, 2015.

Relevant publications

Invited Tutorials

- C. Giatsidis, F. D. Malliaros and M. Vazirgiannis, Graph Mining Tools for Community Detection and Evaluation in Social Networks and the Web. *WWW*, Rome, Italy, 2013.
- C. Giatsidis, F. D. Malliaros and M. Vazirgiannis, Advanced graph mining for community evaluation in social networks and the web, *ACM WSDM*, Rio de Janeiro, Brazil, 2013.
- C. Giatsidis, F. D. Malliaros and M. Vazirgiannis, Community Detection and Evaluation in Social and Information Networks. *WISE*, Thessaloniki, Greece, 2014.
- F. D. Malliaros, M. Vazirgiannis and A.N. Papadopoulos, Core Decomposition: Algorithms and Applications, *IEEE/ACM ASONAM*, Paris, France, 2015.
- F. D. Malliaros, A.N. Papadopoulos, Core Decomposition in Graphs: Concepts, Algorithms and Applications. *ICDM*, Barcelona, 2016.

Demos

http://graphdegeneracy.org/