Network science / Graph mining

Tools & metrics for analyzing a connected world of data

Erwan Le Merrer[1]

[1]Inria, Rennes

ESIR, 2020

Outline

1. Graphs and representations
   - The graph abstraction
2. Classical metrics
   - Preliminaries
   - Metrics
3. Graph data-structures
   - Tools for manipulating graphs
4. Loading a graph with networkx
5. Three important graph models & the web
6. Exploring graphs
7. Importance metrics
8. Community metrics
9. Comparing graphs
10. TVGs: time varying graphs
11. Playing with graphs and Gephi

Outline

1. Graphs and representations
   - The graph abstraction
2. Classical metrics
   - Preliminaries
   - Metrics
3. Graph data-structures
   - Tools for manipulating graphs
4. Loading a graph with networkx
5. Three important graph models & the web
6. Exploring graphs
7. Importance metrics
8. Community metrics
9. Comparing graphs
10. TVGs: time varying graphs
11. Playing with graphs and Gephi

Graphs ?

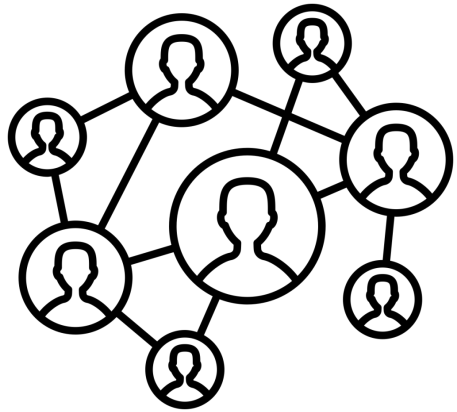
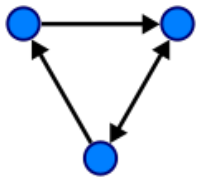


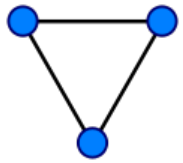
Figure: A graph: entities (nodes) and connections (edges)

- An abstraction for reasoning about characteristics of a relational data, networks...
- Sometimes called “network science”.

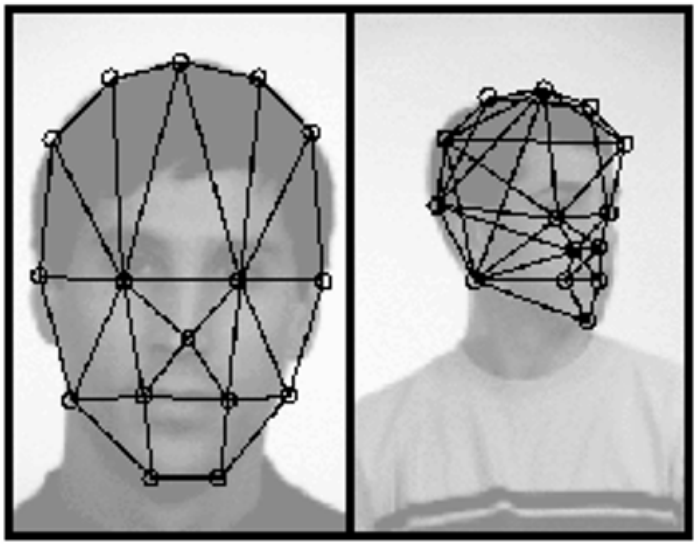
More precisely

- *Vertex*: a graph node, or entity.
- *Edge*: a link connecting two vertices.
- *Directed* and *undirected* graphs:
  - in directed graphs, edges have orientation (arrow end)

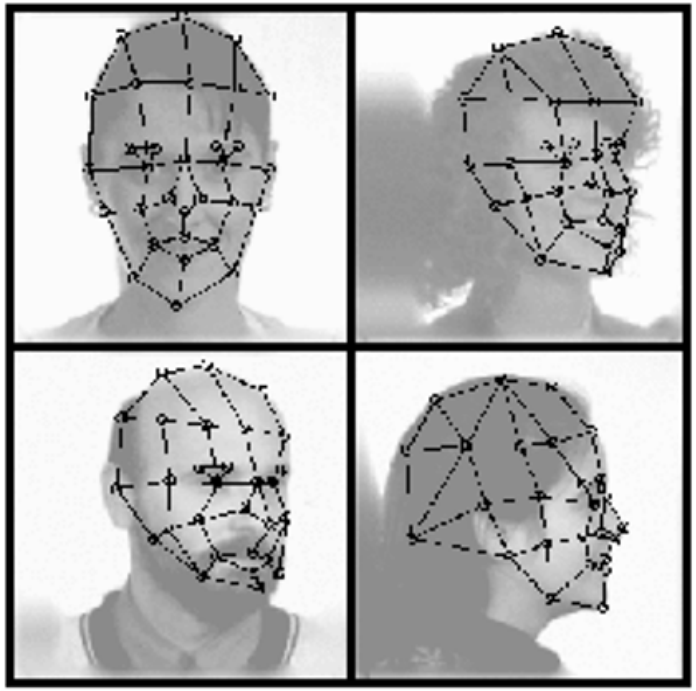




The omnipresence of graphs in applications



grids for face finding



grids for face recognition

Figure: Graphs in computer vision

The omnipresence of graphs in applications

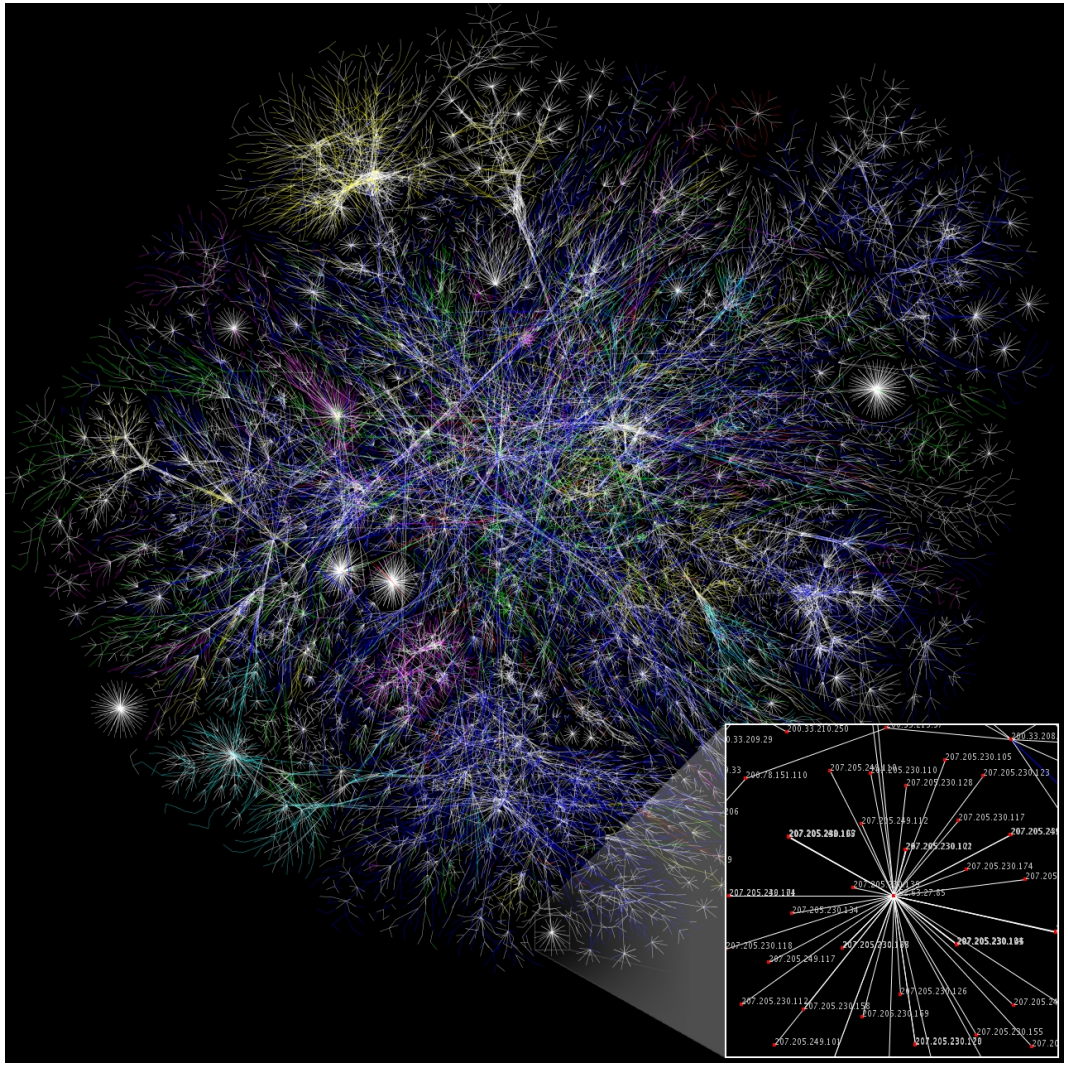


Figure: The Internet AS graph

The omnipresence of graphs in applications

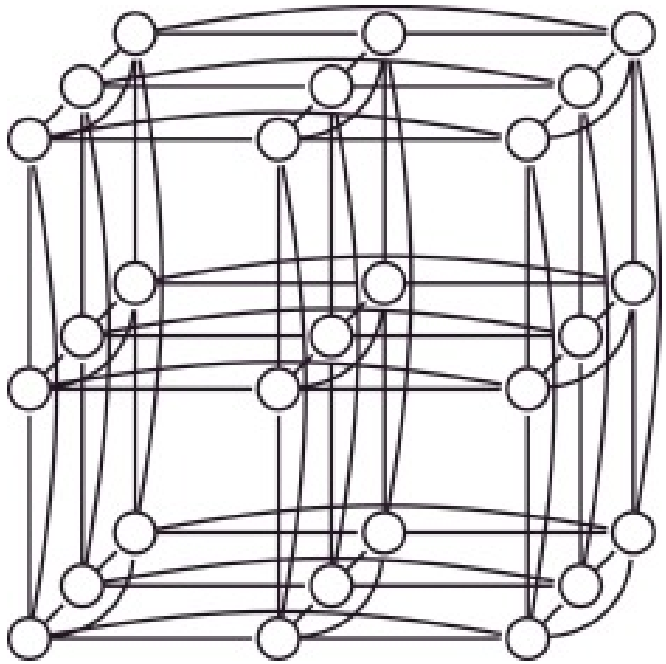


Figure: Interconnecting system-on-chips in a datacenter rack

The omnipresence of graphs in applications

- exemple use in social nets, epidemics...

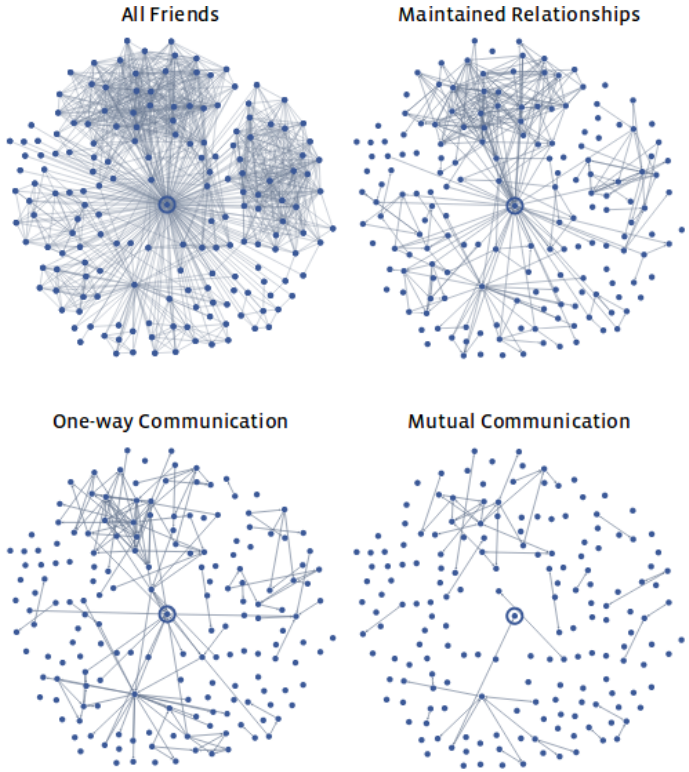


Figure 3.8: Four different views of a Facebook user's network neighborhood, showing the structure of links coresponding respectively to all declared friendships, maintained relationships, one-way communication, and reciprocal (i.e. mutual) communication. (Image from [286].)

Figure: From Networks, Crowds, and Markets: Reasoning about a Highly Connected World . By David Easley and Jon Kleinberg. Cambridge University Press, 2010.

e.g.: recommendations on YouTube

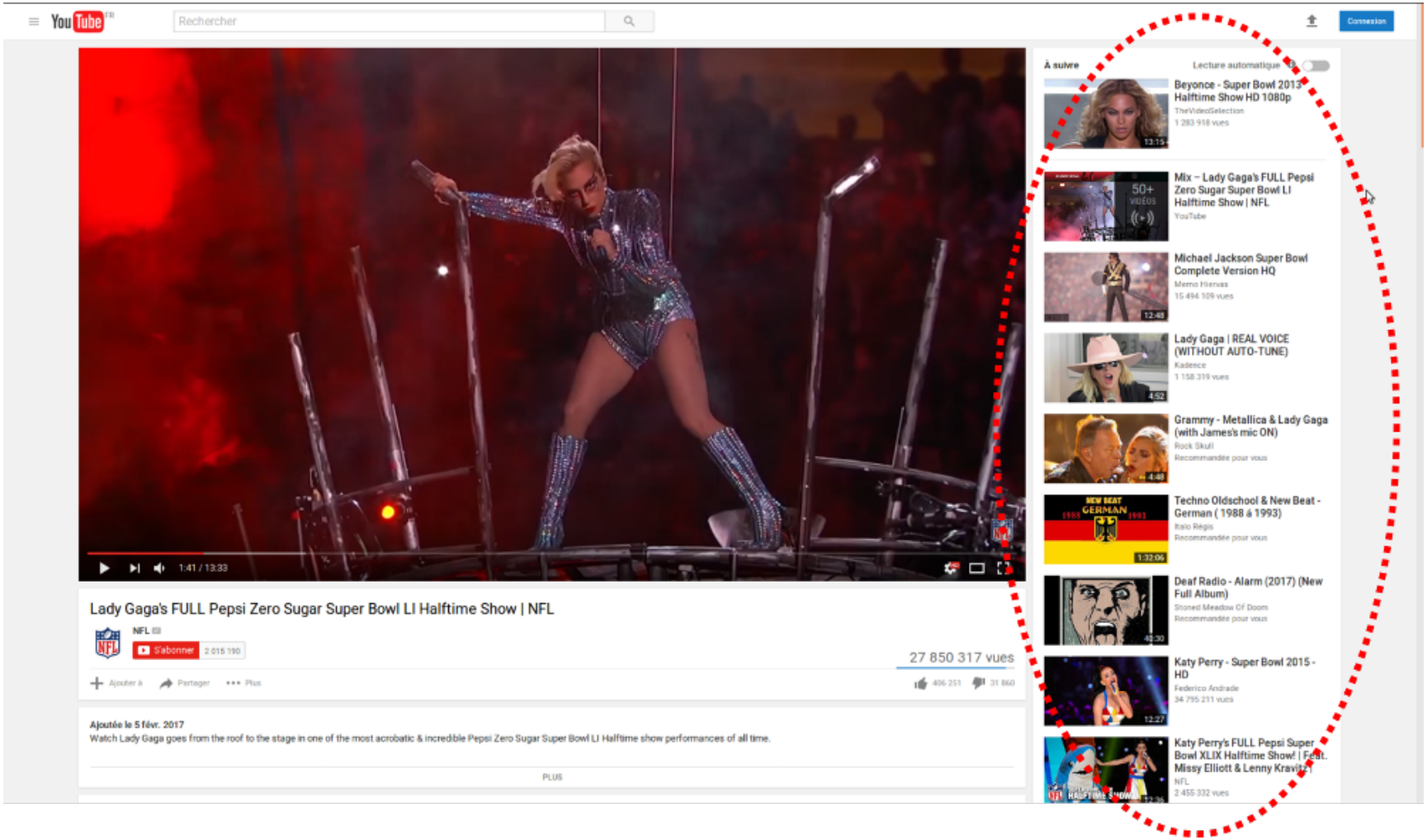
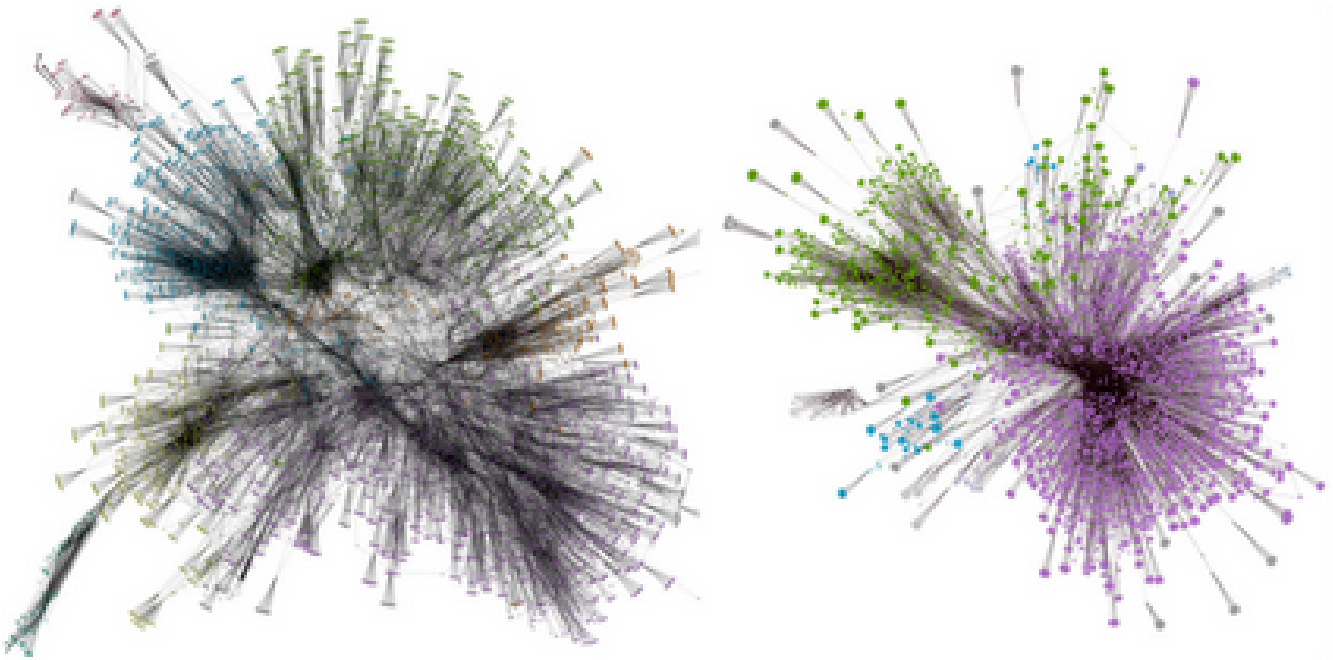


Figure: Recommandations: contextual, personalized?

e.g.: recommendations on YouTube

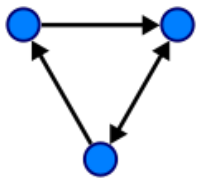


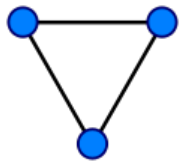
4-hops graphs from a YouTube video, new user (left) and returning user (ri

Figure: Blank profile vs. my recommandations

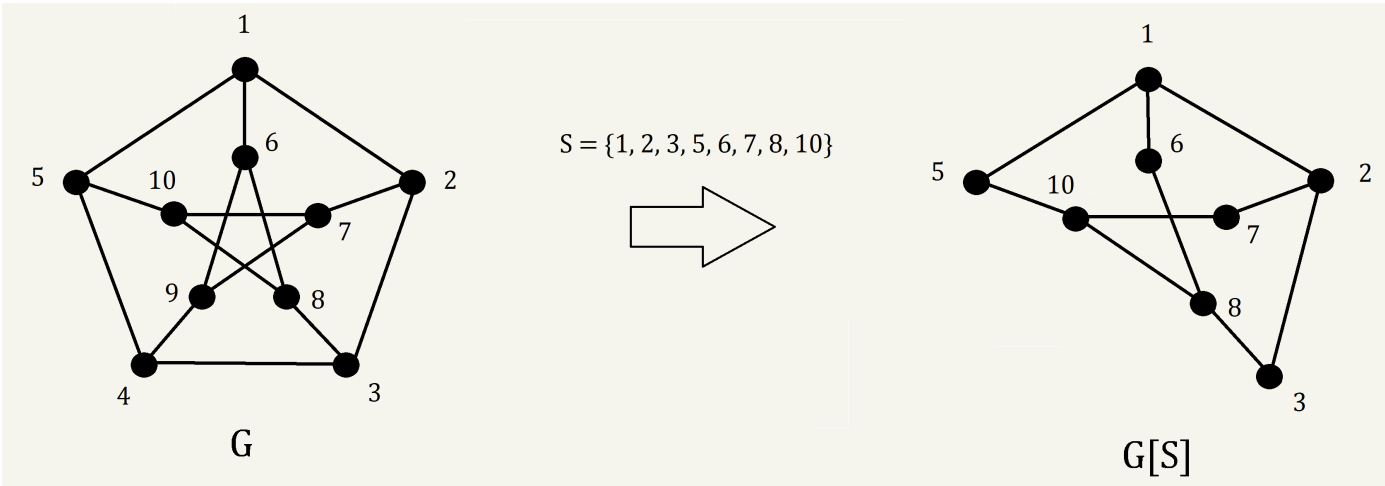
Most important notions

- *Directed* and *undirected* graphs:
  - in directed graphs, edges have orientation (arrow end)



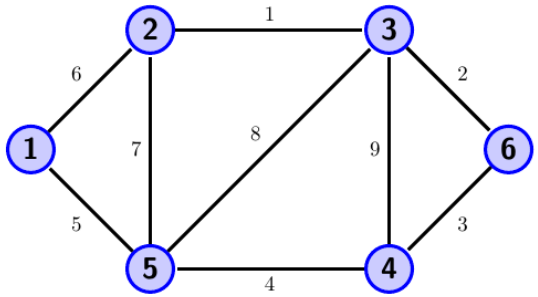


- A *subgraph* of G: formed by a subset of vertices and edges from G.

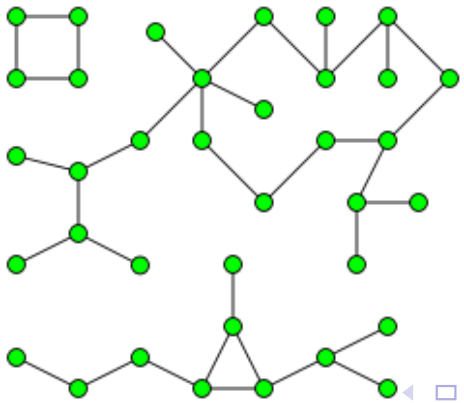


Most important notions (cont'd)

- Edge *weight*: value assigned as a label to an edge.
  - e.g., distance in km of a road from city 1 to 2.

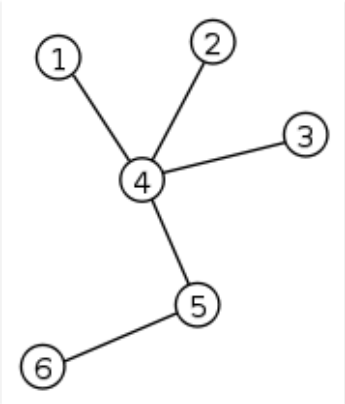


- Graph *connectivity:*
  - A graph is *connected* if there is a *path* btw any pair of vertices.
  - Otherwise, *connected components* are the subgraphs in which paths exist.

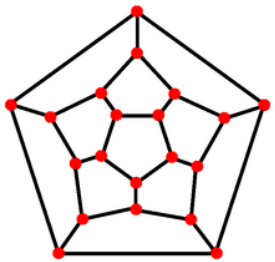


Most important notions (cont'd)

- A *cycle*: a path in which a vertex is reachable from itself.
  - Example of an *acyclic* connected graph: a *tree*



- A *planar* graph: can be draw without any edges crossing each other.



Special *topologies*

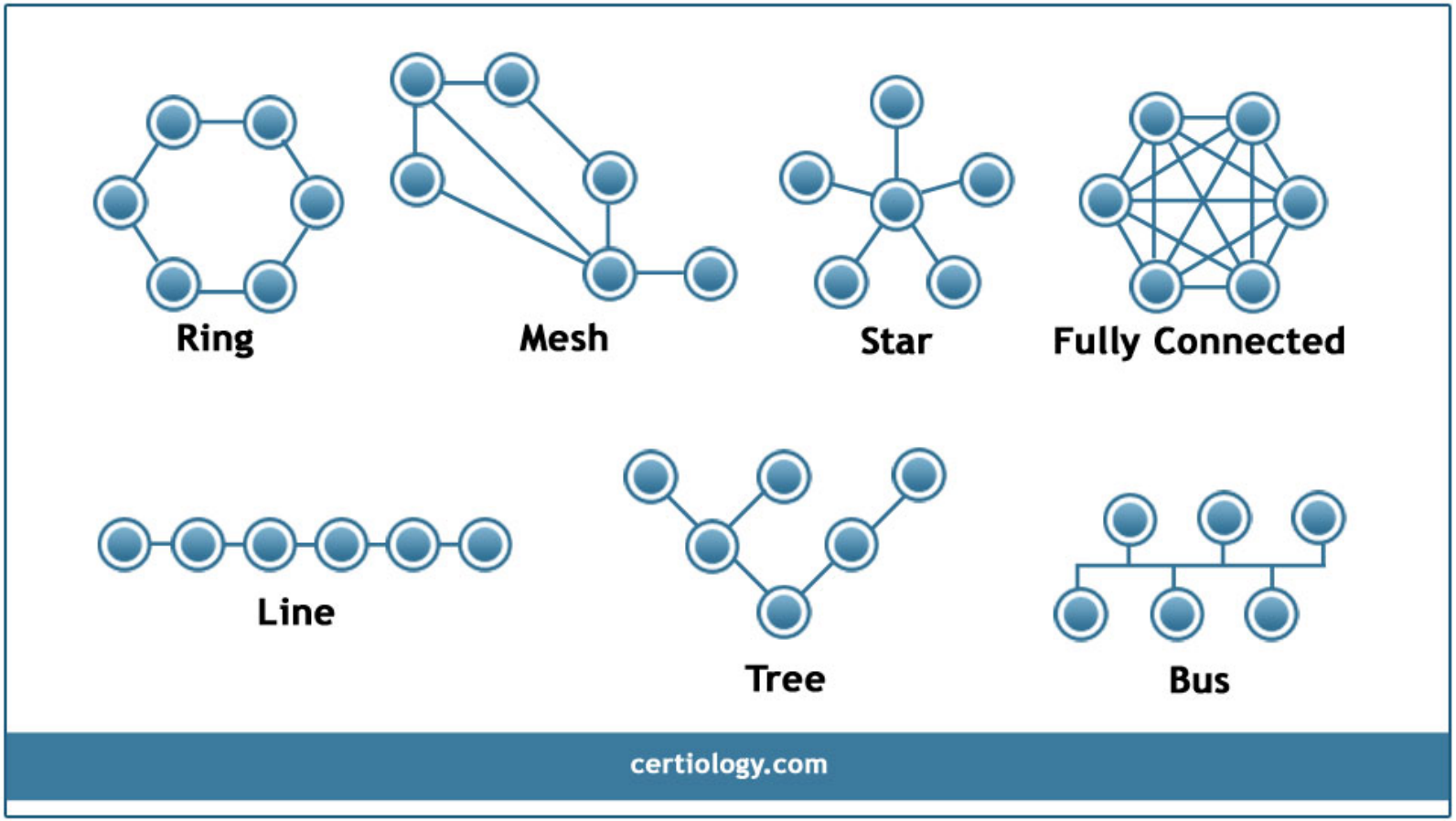
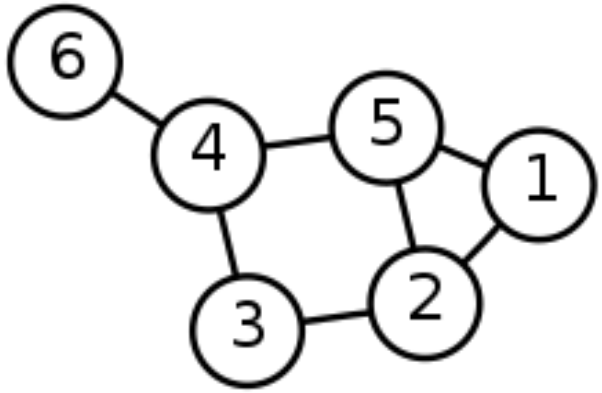


Figure: Graphs to remember, often used as illustrations

Outline

1. Graphs and representations
   - The graph abstraction
2. Classical metrics
   - Preliminaries
   - Metrics
3. Graph data-structures
   - Tools for manipulating graphs
4. Loading a graph with networkx
5. Three important graph models & the web
6. Exploring graphs
7. Importance metrics
8. Community metrics
9. Comparing graphs
10. TVGs: time varying graphs
11. Playing with graphs and Gephi

Basic notations



- $G(V, E)$: graph G with node set V, connected by edge set E.
  - $V = \{1, 2, 3, 4, 5, 6\}$;
    $E = [[1, 5], [1, 2], [2, 3], [2, 5], [3, 4], [4, 5], [4, 6]]$
- Number of nodes is $n = |V|$, edges is $m = |E|$.
- Neighbors of node i are set $\Gamma(i)$.
  - $\Gamma(1) = \{2, 5\}$

Degree of a node

- The degree d_v of node v is equal to $|\Gamma(v)|$ (its number of neighbors).
- Degree span: $0 \leq d_v \leq n-1$ (if no self loops).
- *Degree distribution* P(d) is the probability distribution of each degree in the current graph:

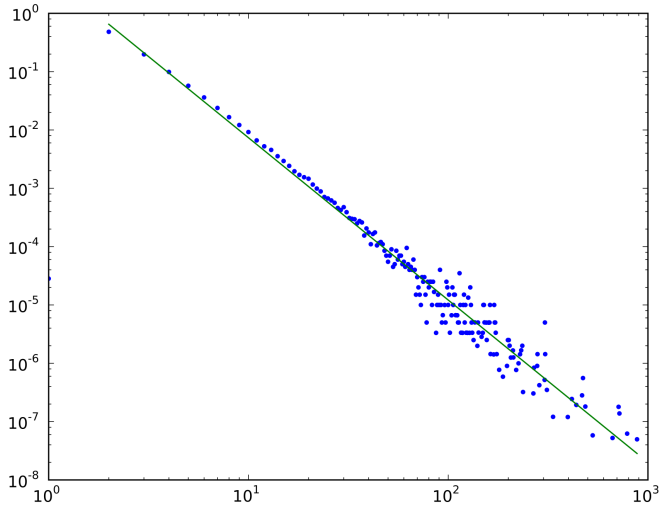


Figure: Degree distribution: x-axis is degree, y-axis is probability

- In(out)-degree of v: counts incoming(outgoing) edges only.

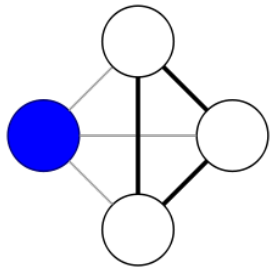
Clustering coefficient

- Every two nodes in a *clique* are neighbors.
- *Local clustering coefficient of a node i measures "how close are $\Gamma(i)$ from being a clique":*

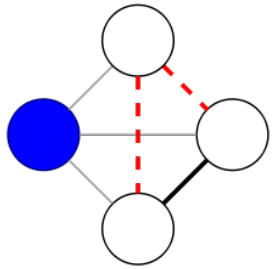
$$C_i = \frac{2|e_{jk} : v_j, v_k \in \Gamma(v_i), e_{jk} \in E|}{d_i(d_i - 1)}$$

- Average clustering coefficient:

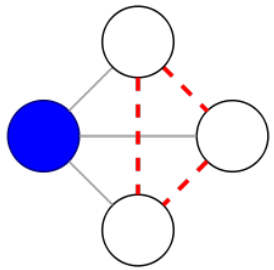
$$\bar{C} = \frac{1}{n} \sum_{i=1}^{n} C_i$$



c = 1



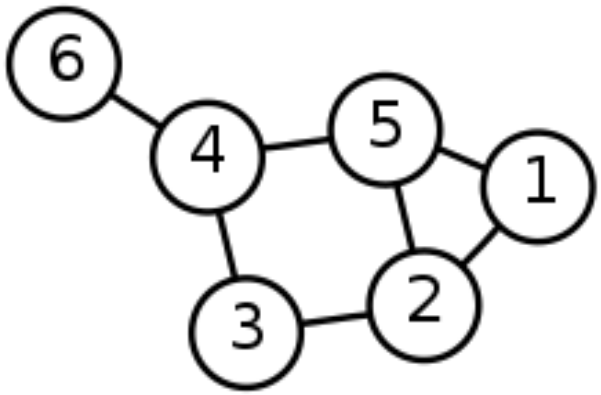
c = 1/3



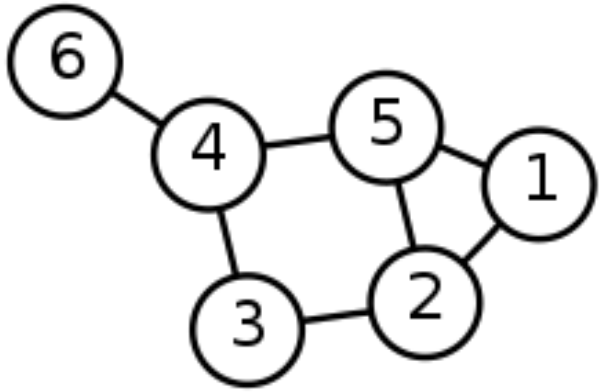
c = 0

Path lengths

- *Path*: sequence of adjacent nodes connecting two nodes (if exists).
  - e.g., two paths btw 6 and 1: (4,5,1) and (4,3,2,5,1).
  - One *hop*: one transition from a node to another.
- *Shortest path*: path of minimal cardinality.
  - Distance $dist(6,1) = |(4,5,1)| = 3$
- *Single-source shortest path* (SSSP): shortest paths from node i to all other nodes ($V \setminus i$).
- *All-pairs shortest paths (APSP)*: SSSP from $\forall i \in V$.

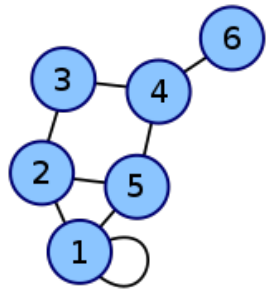


Diameter



- *Average path length*: average of all-pair shortest distances in the graph.
- *Diameter*: longest path of the APSP, i.e., greatest distance between any pair of vertices.
  - $diam(G) = |(4,5,1)| = 3$, starting at node 6.

Algebraic connectivity



- *Degree matrix D*: diagonal matrix containing the degree of each vertex.
- *Adjacency matrix A*: 1 if edge exists (2 for self-loop), 0 otherwise.
- *Laplacian matrix*: $L = D - A$.
- *Algebraic connectivity*: second-smallest eigenvalue of L.
  - $> 0 \iff$ graph is connected.
- Number of 0s as eigenvalues equals number of connected components.

$$\begin{pmatrix} 4 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Figure: D

$$\begin{pmatrix} 2 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

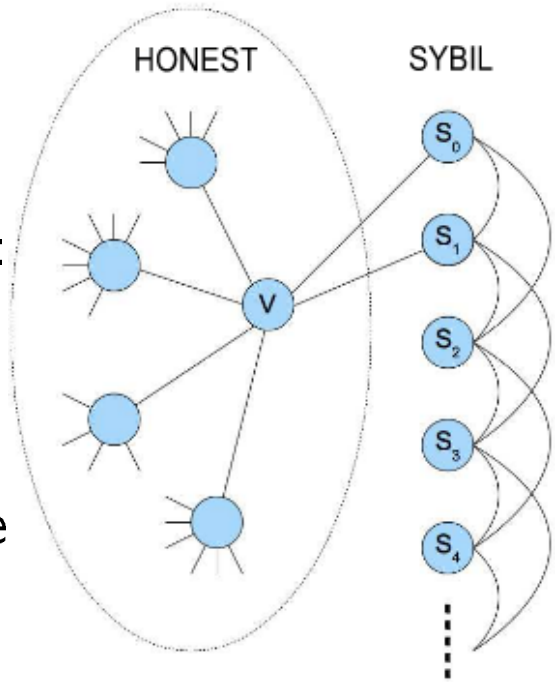
Figure: A

Conductance

- The conductance $\Phi(C)$ of a set C of vertices in a given graph G is the ratio between the number of edges going out from C and the number of edges inside C:

$$\Phi(C) = \frac{|cut(C)|}{vol(C)},$$

where $vol(C)$, is the sum of the degrees of the vertices in C.



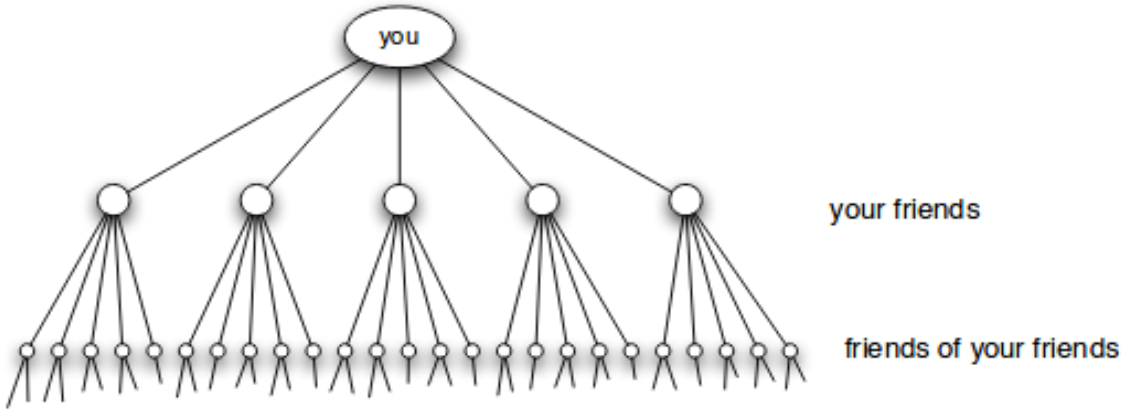
Expansion

- *Expansion* of G: mean number of nodes that are reached in h hops from all nodes:

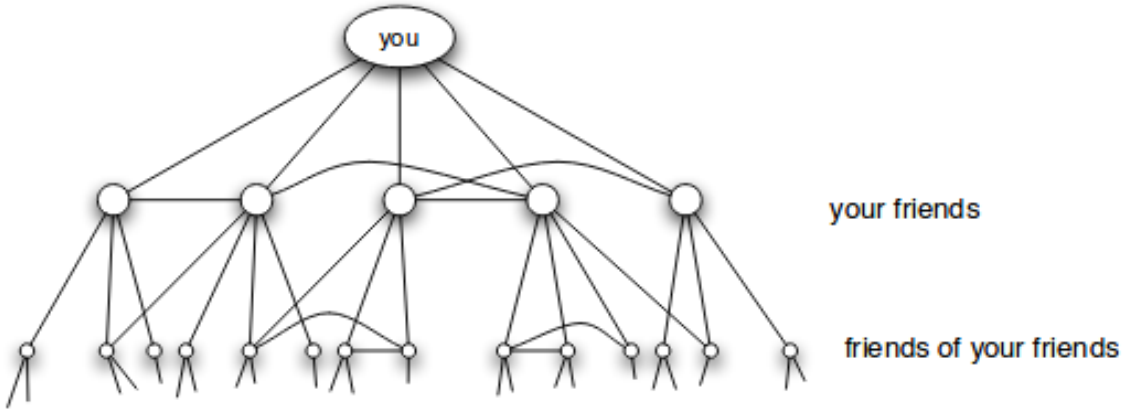
$$e_G(h) = \frac{1}{n^2} \sum_{v \in V} |C_v(h)|,$$

with $C_v(h)$ the set of reachable nodes from v in h hops.

Expansion - example



(a) *Pure exponential growth produces a small world*



(b) *Triadic closure reduces the growth rate*

Figure: Expansion in a social network

Resilience

- Measures the robustness of a graph:

$$r_G(h) = \frac{1}{|E|} \sum_{v \in V} l(v, |C_v(h)|),$$

with $l(v, |C_v(h)|)$ the number of edges that need to be removed to split $C_v(h)$ into 2 sets (of roughly the same size). h: distance (hops).

Outline

1. Graphs and representations
   - The graph abstraction
2. Classical metrics
   - Preliminaries
   - Metrics
3. Graph data-structures
   - Tools for manipulating graphs
4. Loading a graph with networkx
5. Three important graph models & the web
6. Exploring graphs
7. Importance metrics
8. Community metrics
9. Comparing graphs
10. TVGs: time varying graphs
11. Playing with graphs and Gephi

Adjacency list or edge list representations

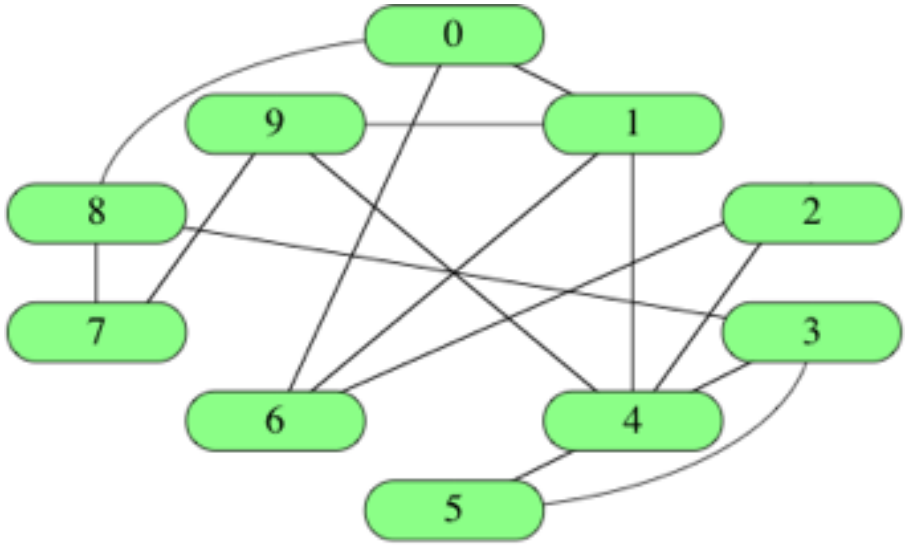


Figure: A graph, to load for analysis

Edge list:

[[0,1], [0,6], [0,8], [1,4], [1,6], [1,9], [2,4], [2,6], [3,4], [3,5], [3,8],[4,5], [4,9], [7,8], [7,9]]

$O(|V|)$ access time to find an edge, but $O(|E|)$ space in memory.

Adjacency list:

[[1, 6, 8], [0, 4, 6, 9], [4, 6], [4, 5, 8], [1, 2, 3, 5, 9], [3, 4], [0, 1, 2], [8, 9], [0, 3, 7], [1, 4, 7]]

$O(1)$ access time to vertex , but $O(|V|)$ to access a given edge.

Image[1]

[1] http://...

Matrix representation

	0	1	2	3	4	5	6	7	8	9
0	0	1	0	0	0	0	1	0	1	0
1	1	0	0	0	1	0	1	0	0	1
2	0	0	0	0	1	0	1	0	0	0
3	0	0	0	0	1	1	0	0	1	0
4	0	1	1	1	0	1	0	0	0	1
5	0	0	0	1	1	0	0	0	0	0
6	1	1	1	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	1	1
8	1	0	0	1	0	0	0	1	0	0
9	0	1	0	0	1	0	0	1	0	0

Figure: Matrix representation of previous graph

Find edge presence in $O(1)$ time, but $\Theta(V^2)$ space in memory.
1's to be replaced by edge weigths for weighted graphs.

Example tool families for manipulating graphs

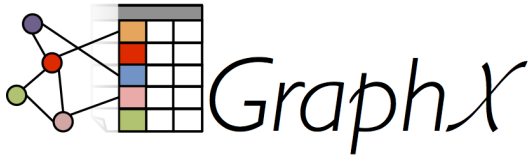


Figure: For massive graphs (cannot fit into on server's memory)

X – Stream

Figure: Big graph processing on a single machine

Figure: For a database-like handling of graphs

NetworkX

Figure: Prototyping in Python, lots of contributions

Outline

1. Graphs and representations
   - The graph abstraction
2. Classical metrics
   - Preliminaries
   - Metrics
3. Graph data-structures
   - Tools for manipulating graphs
4. Loading a graph with networkx
5. Three important graph models & the web
6. Exploring graphs
7. Importance metrics
8. Community metrics
9. Comparing graphs
10. TVGs: time varying graphs
11. Playing with graphs and Gephi

Graph repositories

Stanford Large Network Dataset Collection

- Social networks : online social networks, edges represent interactions between people
- Networks with ground-truth communities : ground-truth network communities in social and information networks
- Communication networks : email communication networks with edges representing communication
- Citation networks : nodes represent papers, edges represent citations
- Collaboration networks : nodes represent scientists, edges represent collaborations (co-authoring a paper)
- Web graphs : nodes represent webpages and edges are hyperlinks
- Amazon networks : nodes represent products and edges link commonly co-purchased products
- Internet networks : nodes represent computers and edges communication
- Road networks : nodes represent intersections and edges roads connecting the intersections
- Autonomous systems : graphs of the internet
- Signed networks : networks with positive and negative edges (friend/foe, trust/distrust)
- Location-based online social networks : Social networks with geographic check-ins
- Wikipedia networks, articles, and metadata : Talk, editing, voting, and article data from Wikipedia
- Temporal networks : networks where edges have timestamps
- Twitter and Memetracker : Memetracker phrases, links and 467 million Tweets
- Online communities : Data from online communities such as Reddit and Flickr
- Online reviews : Data from online review systems such as BeerAdvocate and Amazon

SNAP networks are also availalbe from UF Sparse Matrix collection. **Visualizations of SNAP networks** by Tim Davis.

Social networks

Name	Type	Nodes	Edges	Description
ego-Facebook	Undirected	4,039	88,234	Social circles from Facebook (anonymized)
ego-Gplus	Directed	107,614	13,673,453	Social circles from Google+
ego-Twitter	Directed	81,306	1,768,149	Social circles from Twitter
soc-Epinions1	Directed	75,879	508,837	Who-trusts-whom network of Epinions.com

Figure: Stanford Large Network Dataset Collection, https://snap.stanford.edu/data/

Also: Koblenz Network Collection, http://konect.uni-koblenz.de/
Network Repository, http://networkrepository.com

Outline

1. Graphs and representations
   - The graph abstraction
2. Classical metrics
   - Preliminaries
   - Metrics
3. Graph data-structures
   - Tools for manipulating graphs
4. Loading a graph with networkx
5. Three important graph models & the web
6. Exploring graphs
7. Importance metrics
8. Community metrics
9. Comparing graphs
10. TVGs: time varying graphs
11. Playing with graphs and Gephi

The Erdős–Rényi random graph

- Model $G(n, p)$ for generating a canonical random graph.
  - Create n nodes.
  - Every pair of nodes connected with independant probability p.

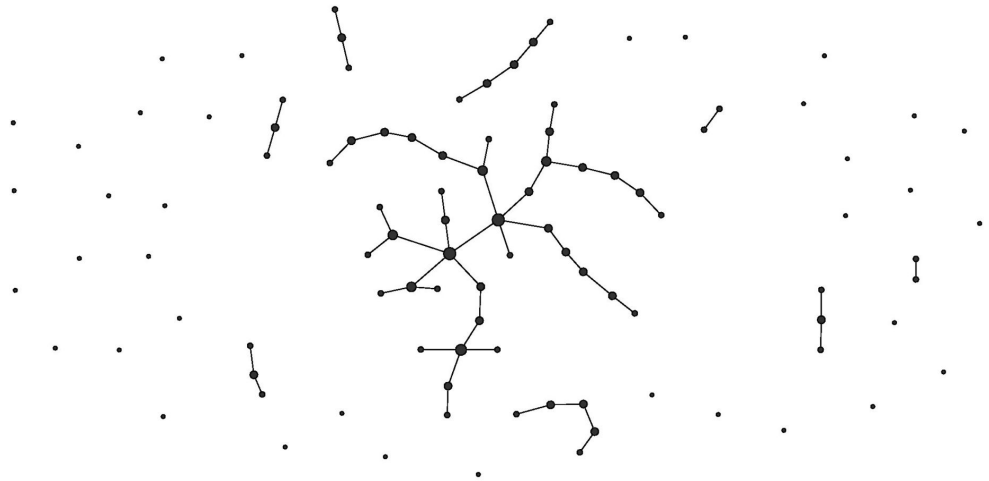


Figure: A random graph with $p = 0.01$.

- If $np = 1$, G almost surely has a largest component of $n = O(n^{2/3})$.
- $p = \frac{\ln n}{n}$ is a threshold for G's connectivity.
- ...

The Watts-Strogatz graph

- Graphs with high clustering (like regular graphs), and low path lengths (like a random graph).
  - Create a ring lattice of n nodes.
  - Replace every edge by a random edge, with probability p.

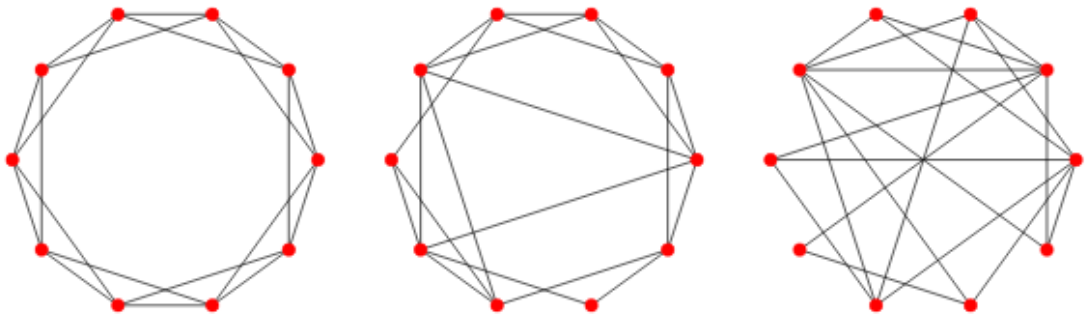


Figure 3.2: WS graphs with $n = 20$, $k = 4$, and $p = 0$ (left), $p = 0.2$ (middle), and $p = 1$ (right).

The Barabási–Albert scale-free graph

- Model to generate a graph with *power-law* degree-distribution.
  - Create m_0 nodes, as a connected graph.
  - Iteratively add one node, and connect it to $m < m_0$ nodes, with probability depending on the degree of existing nodes: $p_i = \frac{d_i}{\sum_j d_j}$ (method called preferential attachment).

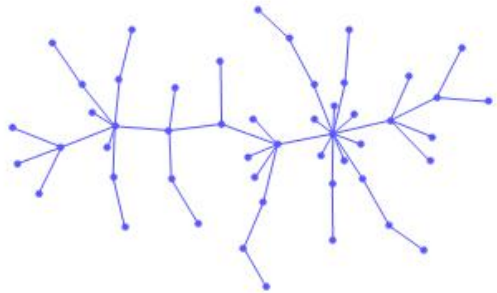


Figure: A Barabási-Albert graph with $n = 50$ and $m_0 = 1$.

- Well connected nodes "accumulate" incoming links: rich gets richer
- Resulting degree distribution is $P(d) \sim d^{-3}$.
- Average path length is $\frac{\ln n}{\ln \ln n}$.

Real structure example

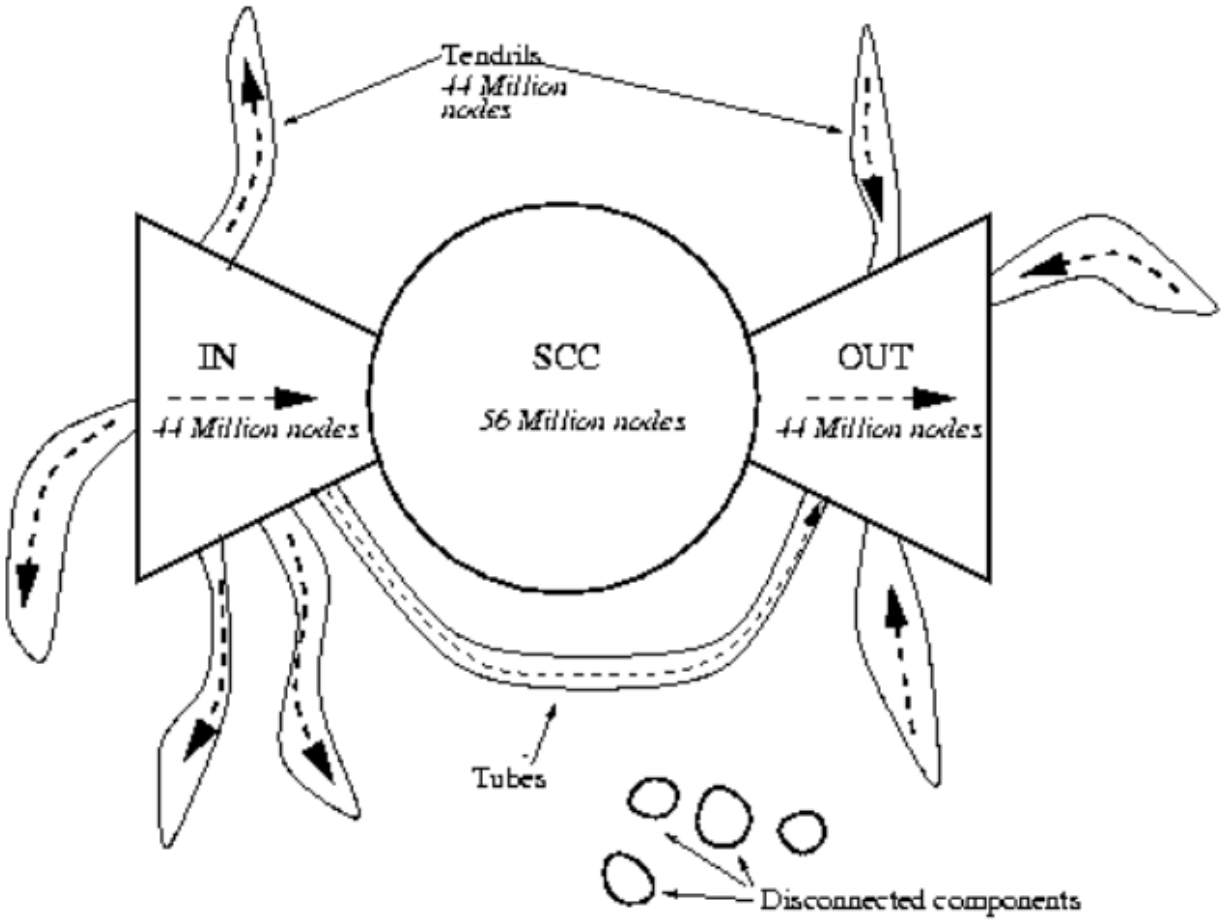
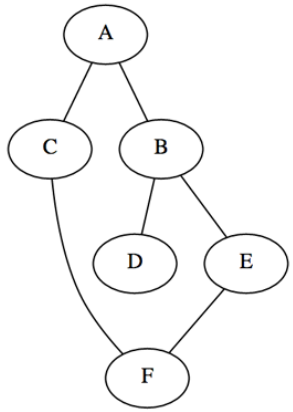


Figure: Box-tie structure of the web

Outline

1. Graphs and representations
   - The graph abstraction
2. Classical metrics
   - Preliminaries
   - Metrics
3. Graph data-structures
   - Tools for manipulating graphs
4. Loading a graph with networkx
5. Three important graph models & the web
6. Exploring graphs
7. Importance metrics
8. Community metrics
9. Comparing graphs
10. TVGs: time varying graphs
11. Playing with graphs and Gephi

Depth first search

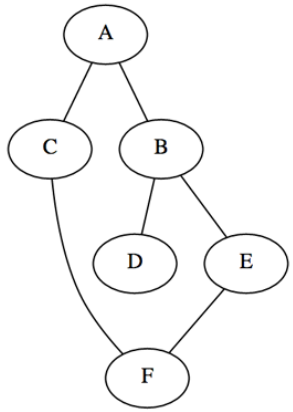


- Graph exploration, from a given start node, *depth* first:

```
def dfs(graph, start):
    visited, stack = set(), [start]
    while stack:
        vertex = stack.pop()
        if vertex not in visited:
            visited.add(vertex)
            stack.extend(graph[vertex] - visited)
    return visited

> dfs(graph, 'A') # {'E', 'D', 'F', 'A', 'C', 'B'}
```

Breadth first search



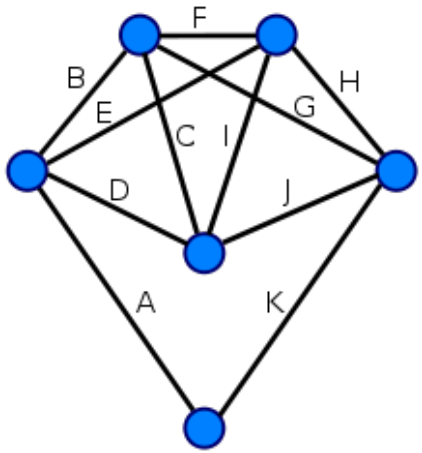
- *breadth* first:

```
def bfs(graph, start):
    visited, queue = set(), [start]
    while queue:
        vertex = queue.pop(0)
        if vertex not in visited:
            visited.add(vertex)
            queue.extend(graph[vertex] - visited)
return visited

>>> bfs(graph, 'A') # {'B', 'C', 'A', 'F', 'D', 'E'}
```

- Queue $\rightarrow$ search in vertices breadth (FIFO)

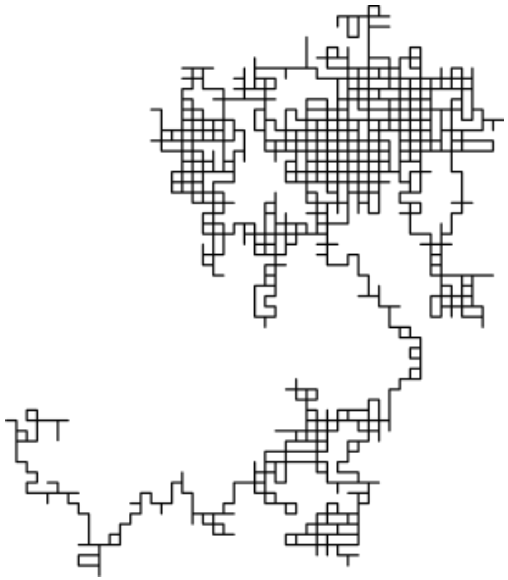
Eulerian path

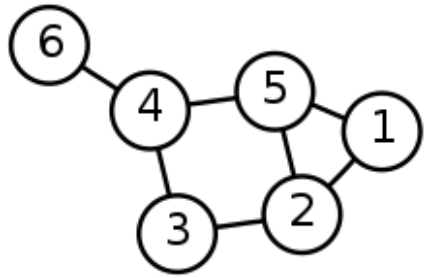


- An Eulerian path visits every edge exactly once (allowing for revisiting vertices).
- Euler's Theorem: A connected graph has an Euler cycle if and only if every vertex has even degree.

Random walk

- Randomized exploration.
- Given a graph and a *start* node, a simple random walk [1] proceeds by random steps:
  - selects uniformly at random a neighbor from walk position
  - jump on it
  - loop process





RDW(6,7hops)=(6,4,3,4,5,4,3,2)

Figure: Random walk on a grid (i.e., 4 neighbors per node)

Random walks - app1: sampling (biased)

- Select a random node in the graph (but biased).
- Given a graph, a *start* node, and a "large" h use a simple random walk:
  - selects uniformly a neighbor; jump on it; $h \leftarrow h-1$
  - loop until $h \leq 0$
- Results in probability of node j to be selected: $P_j = \frac{d_j}{\sum_{i=0}^{n} d_i}$

Random walks - app1: sampling (uniform)

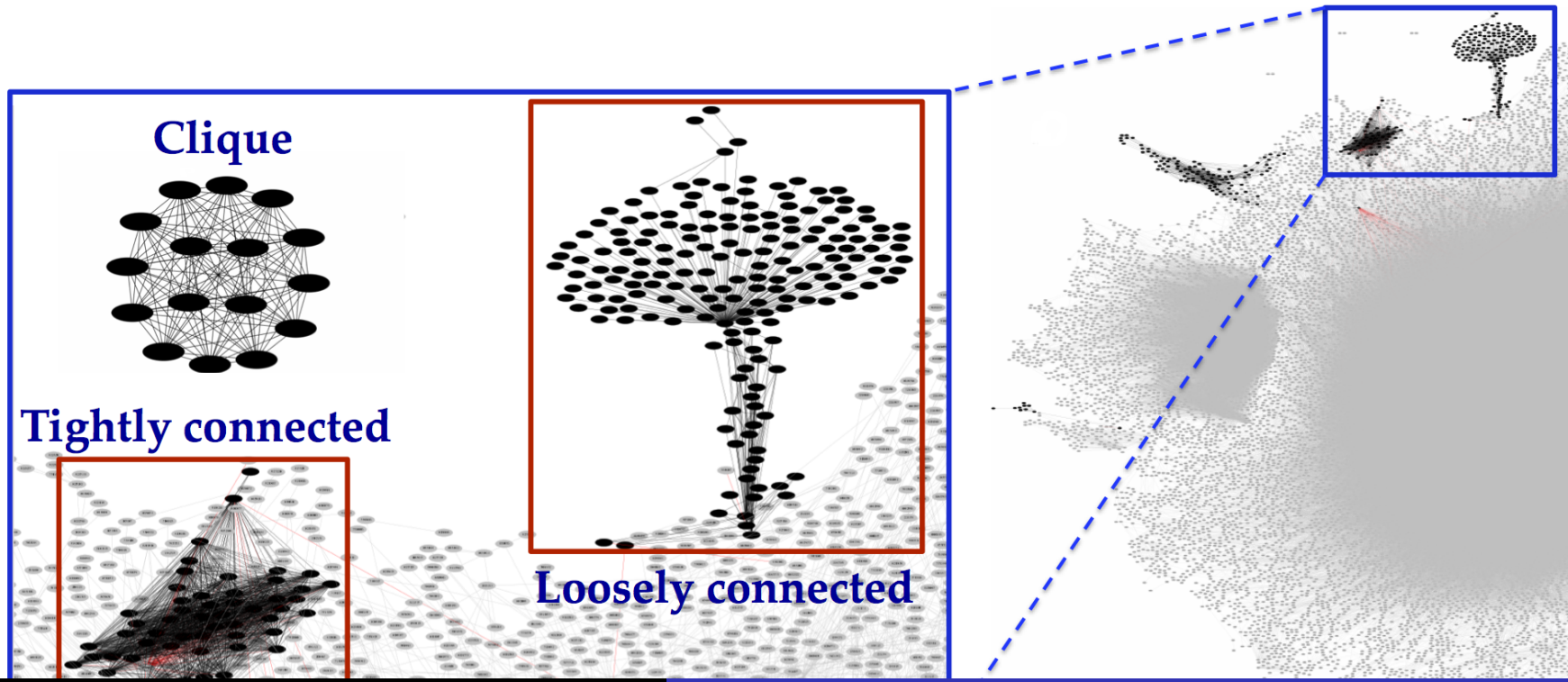
- Select a random node in the graph uniformly (Metropolis-Hastings method).
- Same as for biased except that, from current node i:
  - generate $p \sim U(0,1)$
  - selects uniformly a neighbor j; jump on it if $p \leq \min\{1, \frac{d_i}{d_j}\}$, else stay on i
- Results in probability of node j to be selected: $P_j = \frac{1}{\sum_{i=0}^{n} d_i}$

Random walks - app2: counting

- Distributed computation of n; based on the *birthday paradox* [6].
  - Sample uniformly nodes: $X_{t+1} \leftarrow X_t \cup j$
  - Stop when "collision" after l samples, i.e., when a node j appears twice in X_t
  - $\hat{n} = \sqrt{l^2/2}$

Random walks - app3: sybil detection

- "Early-terminated random walk starting from a non-Sybil node in a social network has a higher degree-normalized (divided by the degree) landing probability to land at a non-Sybil node than a Sybil node". [7]
  - observation holds because the limited number of attack edges forms a narrow passage from the non-Sybil region to the Sybil region in a social network.

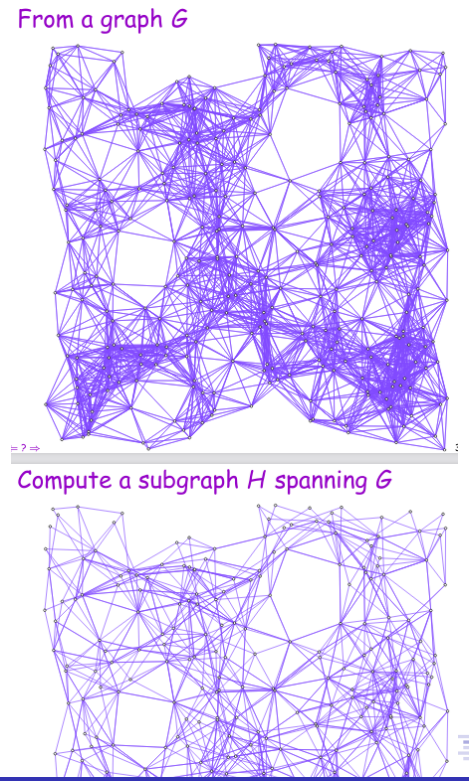


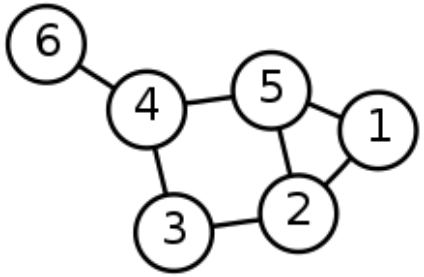
Spanner

- A spanner S of a graph G: subgraph of G with *few* egdes and *short* distances.[2] Tradeoff between number of edges and distance stretch.
- (α, β)-spanner of $G \iff \forall (u, v)$: $dist_H((u, v)) \leq \alpha \times dist_G((u, v)) + \beta$, with α: multiplicative stretch, β: additive stretch.

```
S := {}
For each edge (u,v) in E do
  If dist_H((u,v)) > 2k-1 do
    add (u,v) to S
```

- S is a $(2k-1, 0)$-spanner of G.
- $|V_S| < n_G^{1+1/k}$.





- Epidemic spreading on graph nodes. The SIS model:

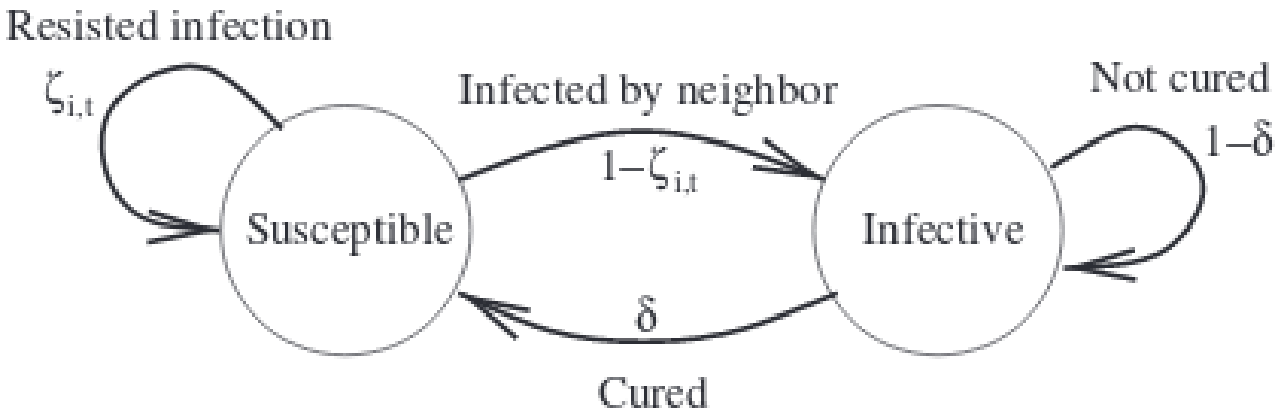


Fig. 1. The SIS model, as seen from a single node. Each node, at each time-step t, is either susceptible (S) or infective (I). A susceptible node i is currently healthy, but can be infected (with probability $1 - \zeta_{i,t}$) by receiving the virus from a neighbor. An infective node can be cured with probability δ; it then goes back to being susceptible. Note that $\zeta_{i,t}$ depends on the both the virus birth rate β and the network topology around node i.