Complex networks
Phys 7682 / CIS 6229: Computational Methods for Nonlinear Systems

- networks are everywhere (and always have been)
  - relationships (edges) among entities (nodes)
- explosion of interest in network structure, function, and evolution over the last decade
  - technology: Internet, World Wide Web
  - biology: genomics, gene expression, protein-protein interactions, physiology
  - sociology: online communities, gossip & rumors, epidemiology, etc.
- interest in mathematical characterization fueled by many common properties among diverse networks
  - degree distributions
  - clustering
  - small-worlds, cliques, and communities
Data structures

- network = graph (a set of nodes connected by edges)
- interested here in **undirected graphs**
  - edge is symmetric between two connecting nodes
- data structures for undirected graph?
  - we use a dictionary of connections
    - dictionary maps *key* to *value*
    - connections[i] = [j0, j1, j2, ...]
    - connections dictionary is directed (asymmetric), so we need to duplicate connections
  - use object-oriented programming to encapsulate internal implementation (dictionary) from external interface (AddNode, AddEdge, etc.)
  - another common implementation is *adjacency matrix*
    - a_{ij} = 1 if nodes i,j connected; 0 otherwise
    - could add method to create adjacency matrix from connection dictionary (hint: use pylab.pcolor to display)
Graph traversal algorithms

- graph traversal
  - iterating through a graph (i.e., over its nodes and edges) and calculating some quantity of interest
    - average shortest path: shortest paths between all pairs of nodes in a graph
    - node and edge betweenness: what fraction of shortest paths each node or edge participates in
    - connected clusters (percolation)
  - traversing nodes and edges, marking nodes as visited so they get visited only once
    - most common: breadth-first and depth-first
- breadth-first search
  - involves iterating through the neighbors of all the nodes in the current shell, and adding to the next shell all subsequent neighbors which have not already been visited
Small-world networks

- motivated by phenomenon of “six degrees of separation”
- studied at Cornell by Duncan Watts and Steve Strogatz
  - Nature 393, 440-442 (1998)
  - simple model of networks with regular short-range bonds and random long-range bonds
  - examination of path lengths and clustering in model and in real-world networks

<table>
<thead>
<tr>
<th>Table 1 Empirical examples of small-world networks</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Network</strong></td>
</tr>
<tr>
<td>Film actors</td>
</tr>
<tr>
<td>Power grid</td>
</tr>
<tr>
<td>C. elegans</td>
</tr>
</tbody>
</table>

Characteristic path length $L$ and clustering coefficient $C$ for three real networks, compared to random graphs with the same number of vertices $n$ and average number of edges per vertex $k$. (Actors: $n = 225,236$, $k = 61$. Power grid: $n = 4,941$, $k = 2.67$. C. elegans: $n = 282$, $k = 14$.) The graphs are defined as follows. Two actors are joined by an edge if they have acted in a film together. We restrict attention to the giant connected component of this graph, which includes ~90% of all actors listed in the Internet Movie Database (available at http://us.imdb.com), as of April 1997. For the power grid, vertices represent generators, transformers and substations, and edges represent high-voltage transmission lines between them. For C. elegans, an edge joins two neurons if they are connected by either a synapse or a gap junction. We treat all edges as undirected and unweighted, and all vertices as identical, recognizing that these are crude approximations. All three networks show the small-world phenomenon: $L > L_{\text{random}}$, but $C < C_{\text{random}}$. 

Decrease in average path length with increasing # of long-range bonds, from Watts & Strogatz.
Percolation

- Some applications
  - flow in porous media (e.g., pressure-driven flow in rock)
  - conductivity of disordered systems (e.g., random resistor networks)
  - forest fires
  - disease transmission in social networks

bond percolation

site percolation
Power laws, correlation lengths, finite-size scaling & universality

Critical points imply scale invariance and power laws

Phase transitions often involve a diverging correlation length $\xi \sim |p-p_c|^{-\nu}$

Diverging correlation length constrained by finite system size $\rightarrow$ finite-size scaling

Microscopically different systems can exhibit the same critical properties $\rightarrow$ universality

$P(p) \sim (p-p_c)^\beta$ \quad probability of being in infinite cluster

$D(s) \sim s^{-T}$ \quad cluster size distribution
Neutral evolutionary networks in biology: what are the mutationally-connected networks of genotypes that produce the same phenotype?

The bow-tie structure of the World Wide Web (SCC = strongly connected component in a directed graph)


Broder et al., Computer Networks (2000)
Connected components & computational sustainability

Wildlife corridors to allow for animal migration

Given a graph $G$ with a set of reserves:

Find a sub-graph of $G$ that:

- contains the reserves
- is connected
- with cost below a given budget
- and, has maximum utility

from Carla Gomes
Types of network structures

- **Undirected graphs**
  - edges symmetric between nodes
    - e.g., protein-protein interaction network: what interacts with what?
    - e.g., scientific collaboration network: who has written a paper together?

- **Directed graphs**
  - edges directed from a source node to destination node
    - e.g., web pages: what links to what?
    - e.g., food webs: who eats whom?
    - e.g., river networks: what flows where?
    - e.g., software systems: what classes contain other classes?
Types of network structures

• Weighted graphs
  - edge-weighted
    ‣ e.g., traffic flow: traffic capacity and/or travel time of each road/edge (directed)
    ‣ e.g., protein-protein interaction network: probability of two proteins interacting (undirected)
  - node-weighted
    ‣ e.g., cost of land parcels for grizzly bear migration

• Multipartite graphs
  - e.g., bipartite graph of places and transitions in a Petri net
  - e.g., actors and movies in an actor collaboration network
Network growth, structure, etc.

- Other papers/projects for further consideration (or maybe you have your own in mind)
  - Barabasi and Albert, “Emergence of scaling in random networks”
    - power-law degree distributions (actor network with bipartite graph?)
  - Callaway et al., “Are randomly grown graphs really random?”
    - essential singularity for onset of connected cluster
  - Girvan and Newman, “Community structure in social and biological networks”
    - quantifying tightly-knit groups in large networks
  - Yu et al., “The importance of bottlenecks in protein networks: Correlation with gene essentiality and expression dynamics”
    - role of betweenness in organizing biological networks
  - Kaiser and Hilgetag, “Nonoptimal Component Placement, but Short Processing Paths, due to Long-Distance Projections in Neural Systems”
    - investigation of wiring lengths and processing paths from neural network data
  - graph layout is also an interesting problem
    - how to optimally place graph nodes and edges (e.g., on a 2D display) when there is no intrinsic geometric information attached to graph
NetworkX: a Python package for creating, manipulating, and analyzing networks (networkx.lanl.gov)

NetworkX

High productivity software for complex networks

About

NetworkX (NX) is a Python package for the creation, manipulation, and study of the structure, dynamics, and functions of complex networks.

Features:

- Includes standard graph-theoretic and statistical physics functions
- Easy exchange of network algorithms between applications, disciplines, and platforms
- Includes many classic graphs and synthetic networks
- Nodes and edges can be "anything" (e.g. time-series, text, images, XML records)
- Exploits existing code from high-quality legacy software in C, C++, Fortran, etc.
- Open source (encourages community input)
- Unit-tested

Additional benefits due to Python:

- Allows fast prototyping of new algorithms
- Easy to teach
- Multi-platform
- Allows easy access to almost any database

Quick Example

Just write in Python

```python
>>> import networkx as NX
>>> G=NX.Graph()
>>> G.add_edge(1,2)
>>> G.add_node("spam")
>>> print G.nodes()
[1, 2, 'spam']
>>> print G.edges()
[(1, 2)]
```