open.michigan

UNIVERSITY OF MICHIGAN

School of Information
University of Michigan

Network basics & some tools

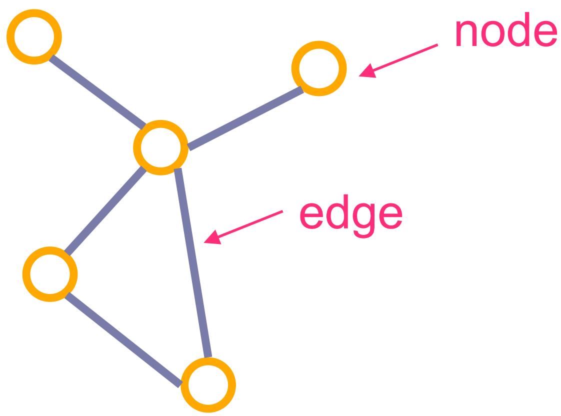
Lada Adamic

Outline

- What is a network?
  - a bunch of nodes and edges
- How do you characterize it?
  - with some basic network metrics
- How did network analysis get started
  - it was the mathematicians
- How do you analyze networks today?
  - with pajek or other software

What are networks?

- Networks are collections of points joined by lines.



"Network" ≡ "Graph"

points	lines	
vertices	edges, arcs	math
nodes	links	computer science
sites	bonds	physics
actors	ties, relations	sociology

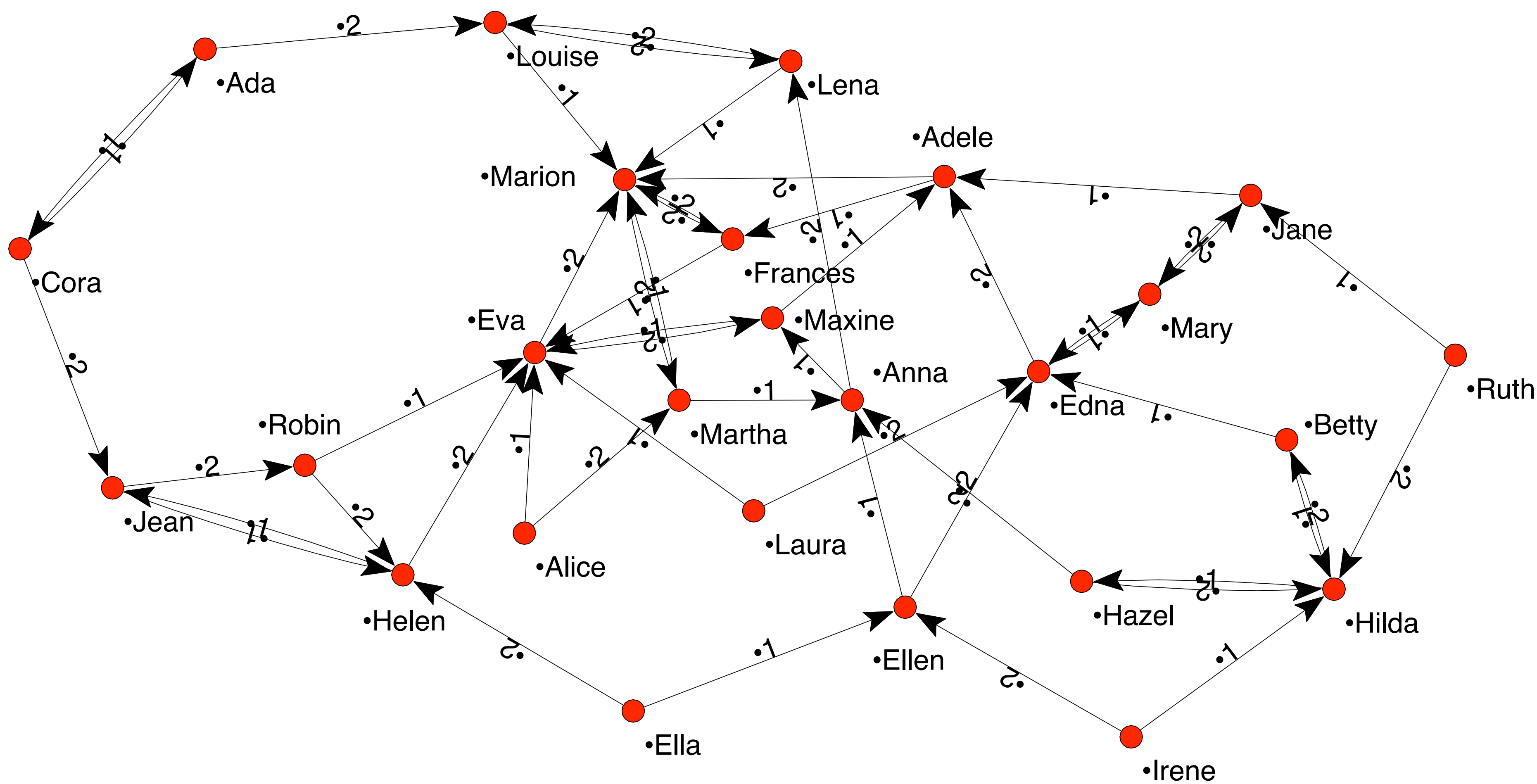
Network elements: edges

- Directed (also called arcs)
  - A -> B
    - A likes B, A gave a gift to B, A is B’s child
- Undirected
  - A <-> B or A – B
    - A and B like each other
    - A and B are siblings
    - A and B are co-authors

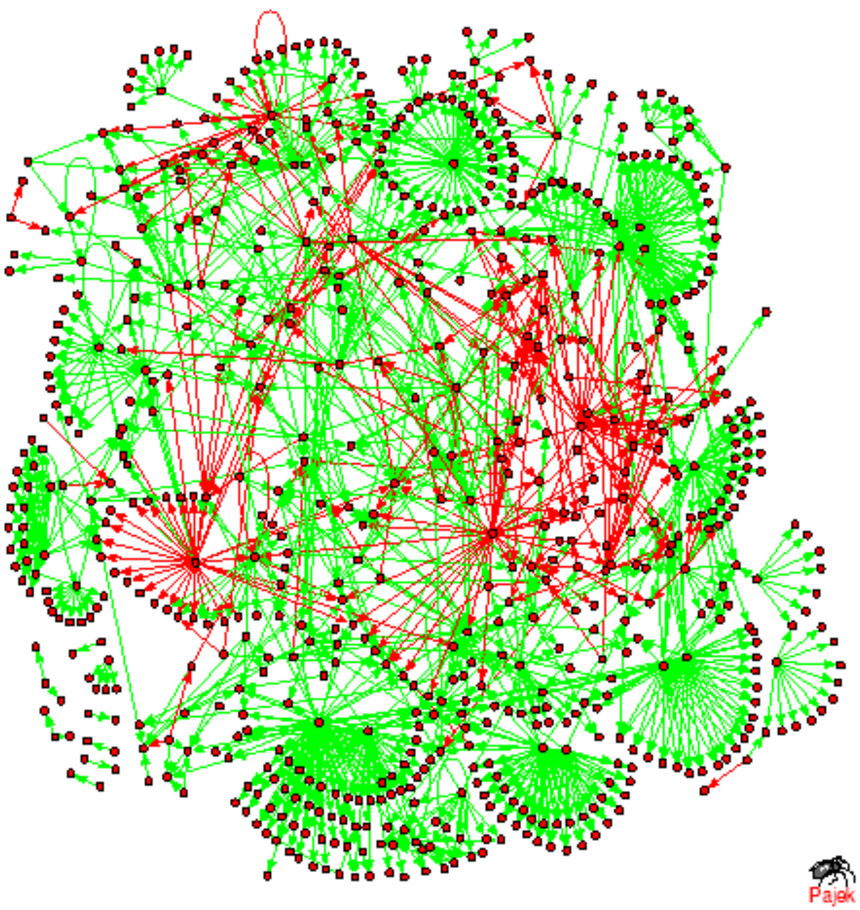
- Edge attributes
  - weight (e.g. frequency of communication)
  - ranking (best friend, second best friend…)
  - type (friend, relative, co-worker)
  - properties depending on the structure of the rest of the graph: e.g. betweenness

Directed networks

- girls' school dormitory dining-table partners (Moreno, *The sociometry reader*, 1960)
- first and second choices shown



Edge weights can have positive or negative values



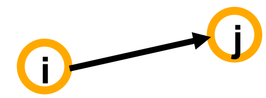
•Transcription regulatory network in baker's yeast

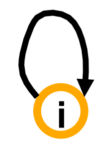
- One gene activates/ inhibits another
- One person trusting/ distrusting another
  - Research challenge: How does one 'propagate' negative feelings in a social network? Is my enemy's enemy my friend?

Source: undetermined

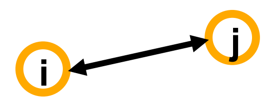
Adjacency matrices

- Representing edges (who is adjacent to whom) as a matrix
  - A_{ij} = 1 if node i has an edge to node j
    = 0 if node i does not have an edge to j



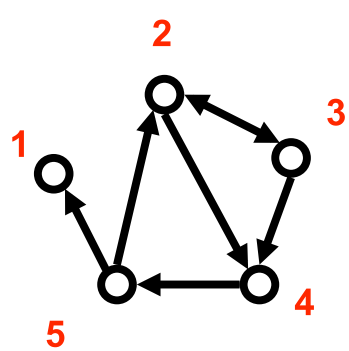


  - A_{ii} = 0 unless the network has self-loops



  - $A_{ij} = A_{ji}$ if the network is undirected, or if i and j share a reciprocated edge

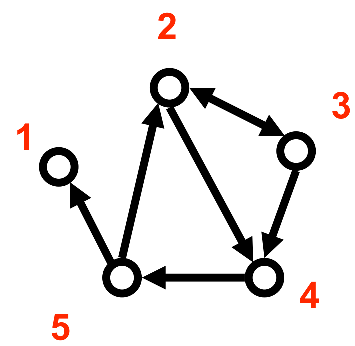
•Example:



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

Adjacency lists

- Edge list
  - 2 3
  - 2 4
  - 3 2
  - 3 4
  - 4 5
  - 5 2
  - 5 1

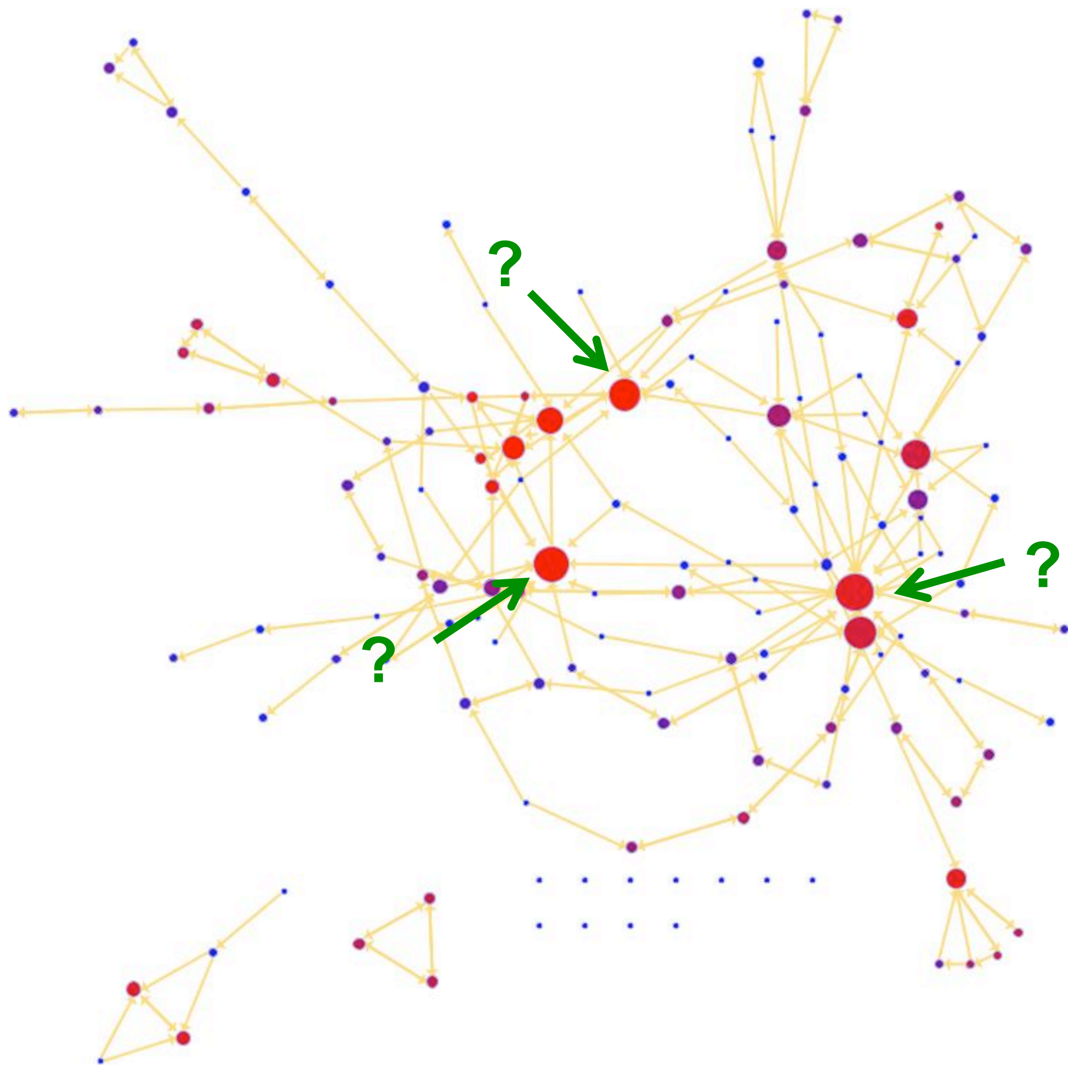


- Adjacency list
  - is easier to work with if network is
    - large
    - sparse
  - quickly retrieve all neighbors for a node
    - 1:
    - 2: 3 4
    - 3: 2 4
    - 4: 5
    - 5: 1 2

Outline

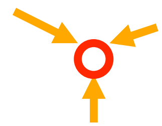
- What is a network?
  - a bunch of nodes and edges
- How do you characterize it?
  - with some basic network metrics
- How did network analysis get started
  - it was the mathematicians
- How do you analyze networks today?
  - with pajek or other software

Characterizing networks:
Who is most central?



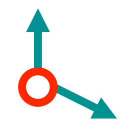
Nodes

- Node network properties
  - from immediate connections
    - indegree
      how many directed edges (arcs) are incident on a node



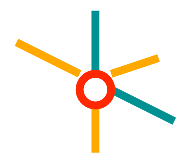
indegree=3

    - outdegree
      how many directed edges (arcs) originate at a node



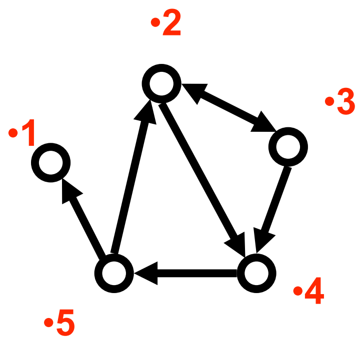
outdegree=2

    - degree (in or out)
      number of edges incident on a node



degree=5

  - from the entire graph
    - centrality (betweenness, closeness)



Node degree from matrix values

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

•example: outdegree for node 3 is 2, which we obtain by summing the number of non-zero entries in the 3rd *row*

$$\sum_{j=1}^{n} A_{3j}$$

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

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

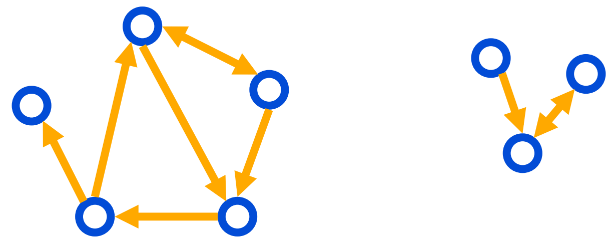
•example: the indegree for node 3 is 1, which we obtain by summing the number of non-zero entries in the 3rd *column*

$$\sum_{i=1}^{n} A_{i3}$$

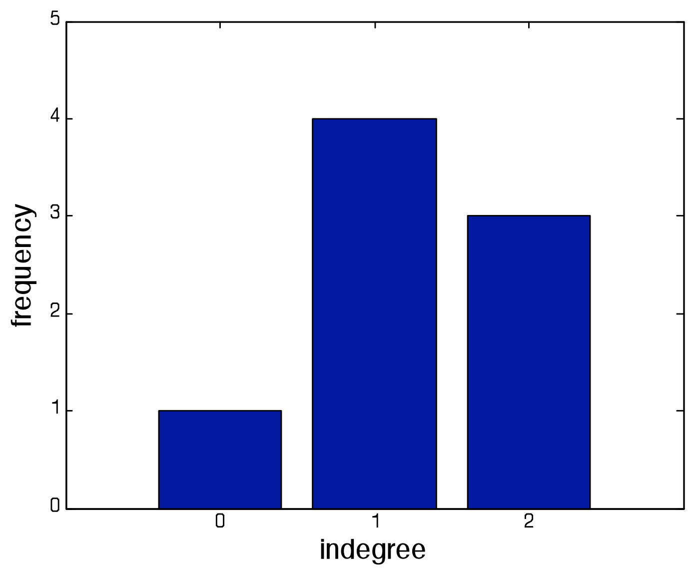
•A = $\begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 \end{pmatrix}$

Network metrics: degree sequence and degree distribution

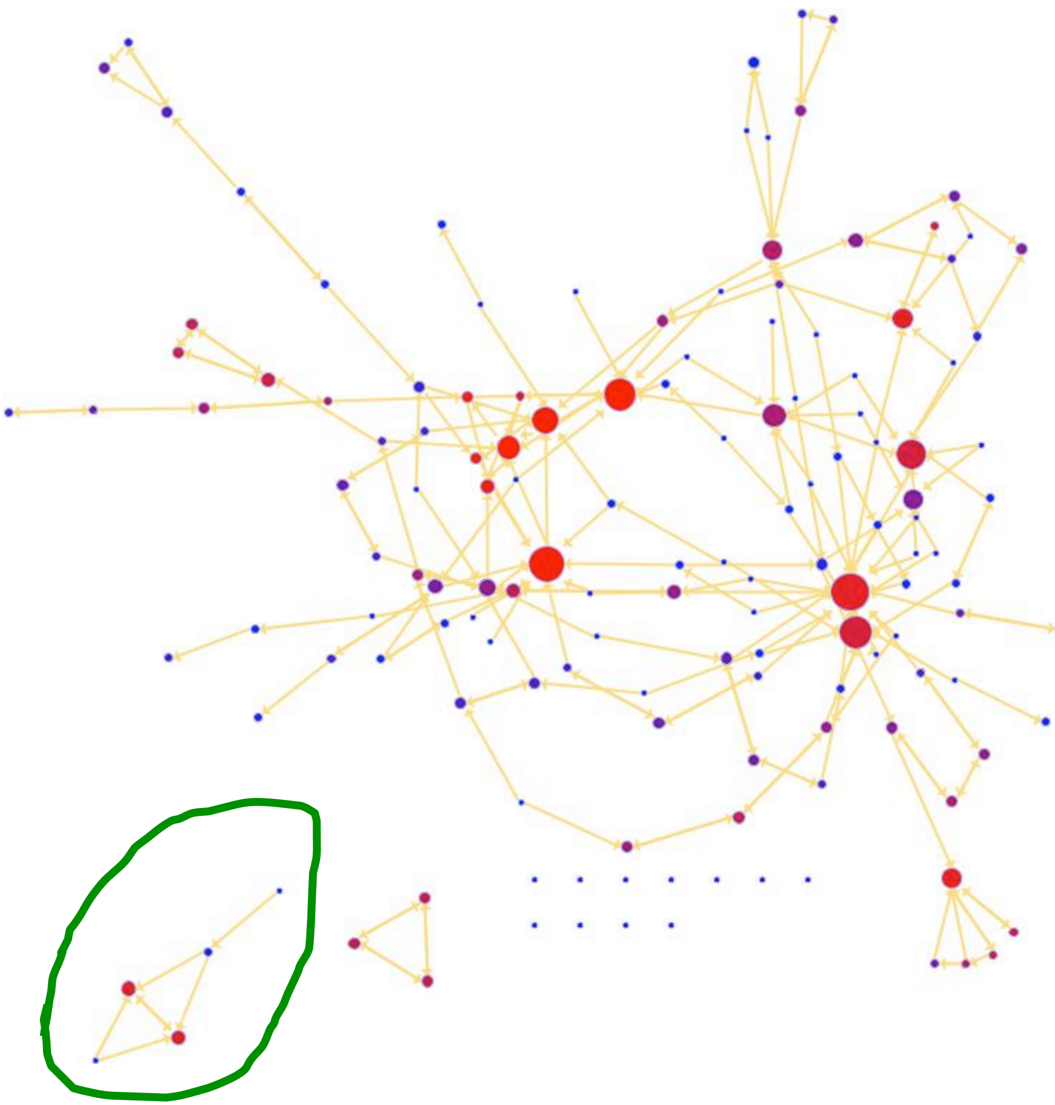
- Degree sequence: An ordered list of the (in,out) degree of each node
  - In-degree sequence:
    - [2, 2, 2, 1, 1, 1, 1, 0]
  - Out-degree sequence:
    - [2, 2, 2, 2, 1, 1, 1, 0]
  - (undirected) degree sequence:
    - [3, 3, 3, 2, 2, 1, 1, 1]



- Degree distribution: A frequency count of the occurrence of each degree
  - In-degree distribution:
    - [(2,3) (1,4) (0,1)]
  - Out-degree distribution:
    - [(2,4) (1,3) (0,1)]
  - (undirected) distribution:
    - [(3,3) (2,2) (1,3)]

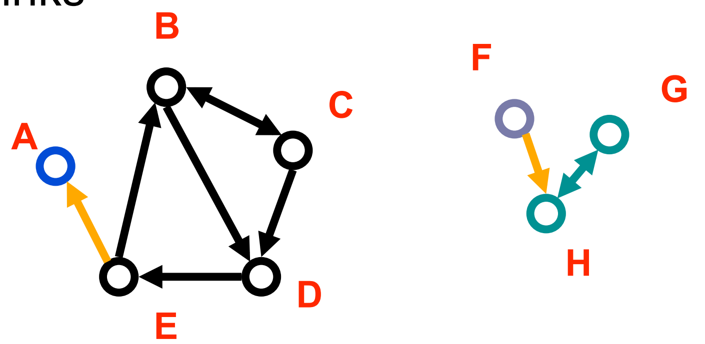


Characterizing networks:
Is everything connected?

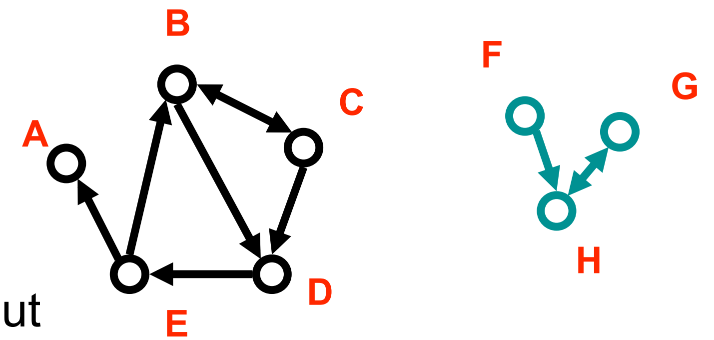


Network metrics: connected components

- Strongly connected components
  - Each node within the component can be reached from every other node in the component by following directed links
  - Strongly connected components
    - B C D E
    - A
    - G H
    - F



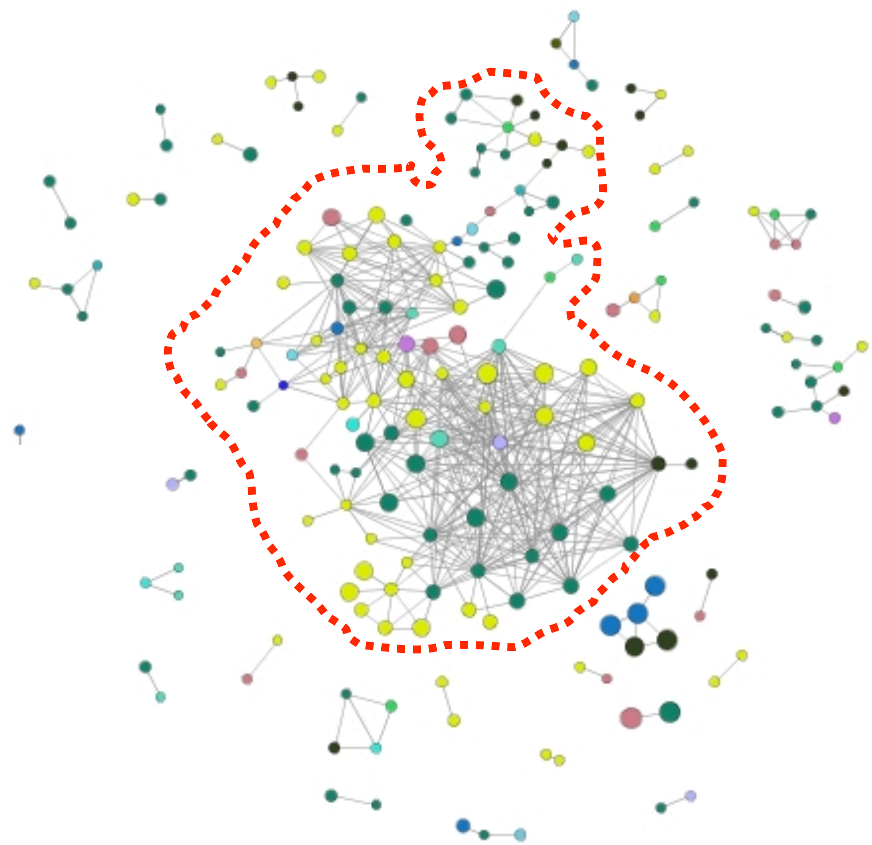
- Weakly connected components: every node can be reached from every other node by following links in either direction
  - Weakly connected components
    - A B C D E
    - G H F



- In undirected networks one talks simply about ‘connected components’

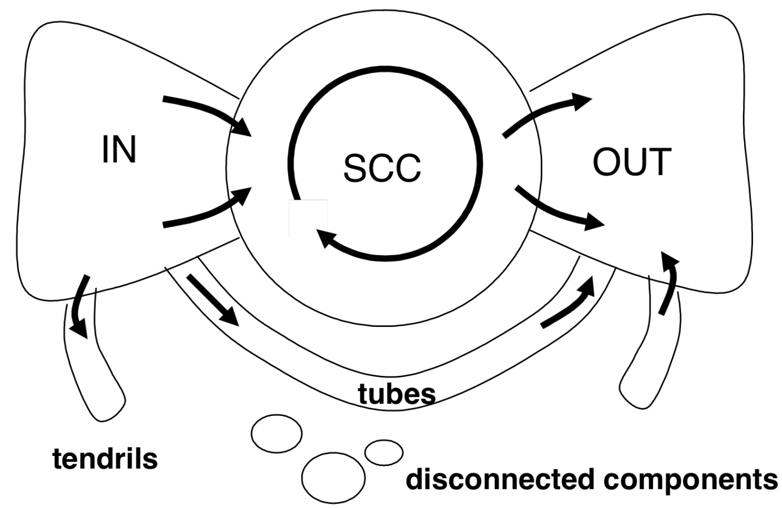
network metrics: size of giant component

- if the largest component encompasses a significant fraction of the graph, it is called the **giant component**

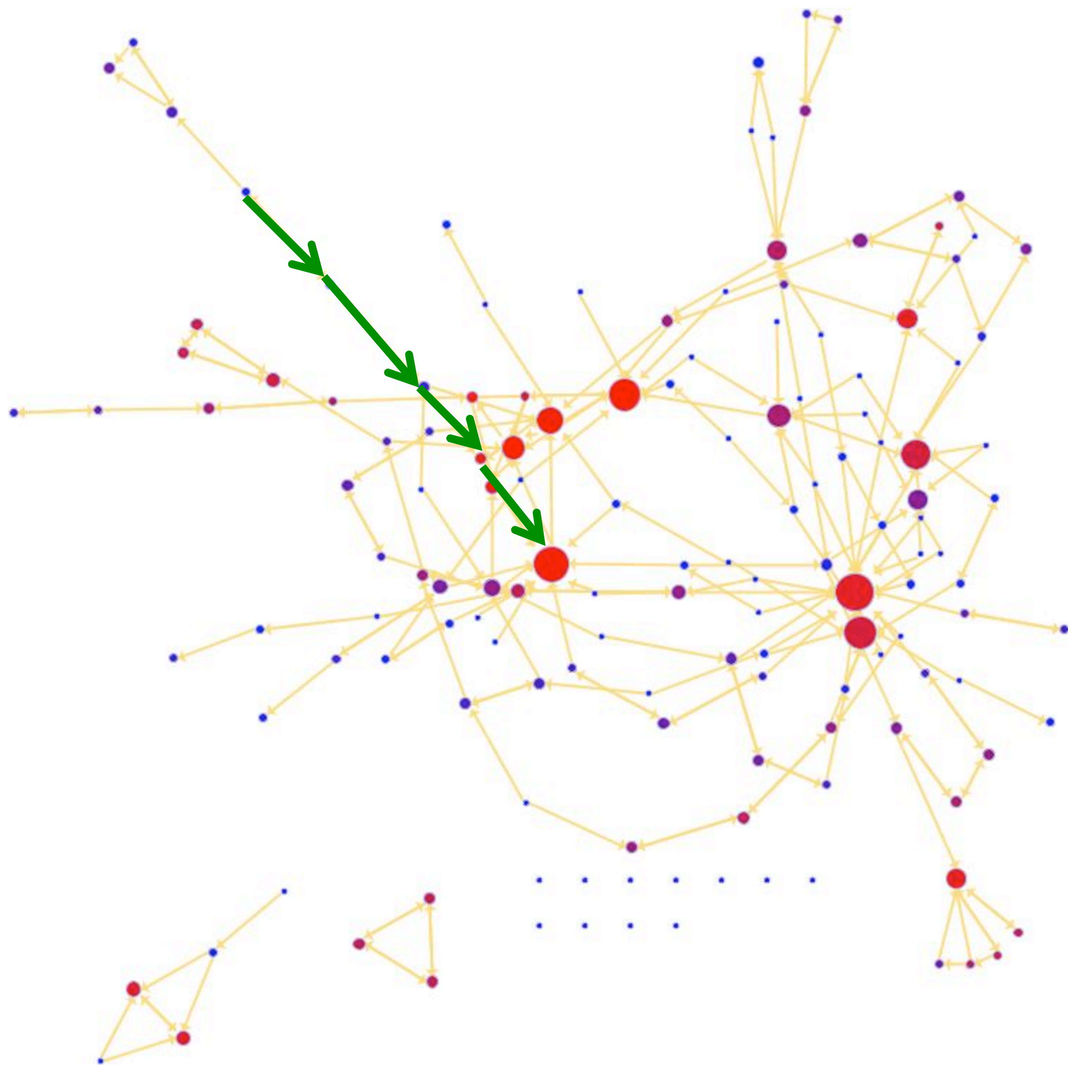


network metrics: bowtie model of the web

- The Web is a directed graph:
  - webpages link to other webpages
- The connected components tell us what set of pages can be reached from any other just by surfing (no ‘jumping’ around by typing in a URL or using a search engine)
- Broder et al. 1999 – crawl of over 200 million pages and 1.5 billion links.
- SCC – 27.5%
- IN and OUT – 21.5%
- Tendrils and tubes – 21.5%
- Disconnected – 8%

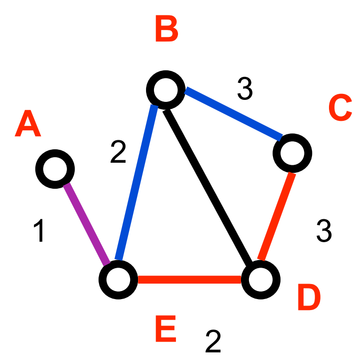


Characterizing networks:
How far apart are things?



Network metrics: shortest paths

- Shortest path (also called a geodesic path)
  - The shortest sequence of links connecting two nodes
  - Not always unique

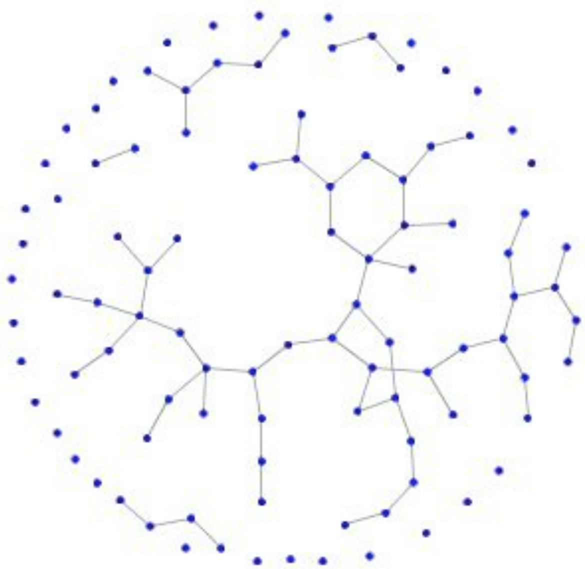


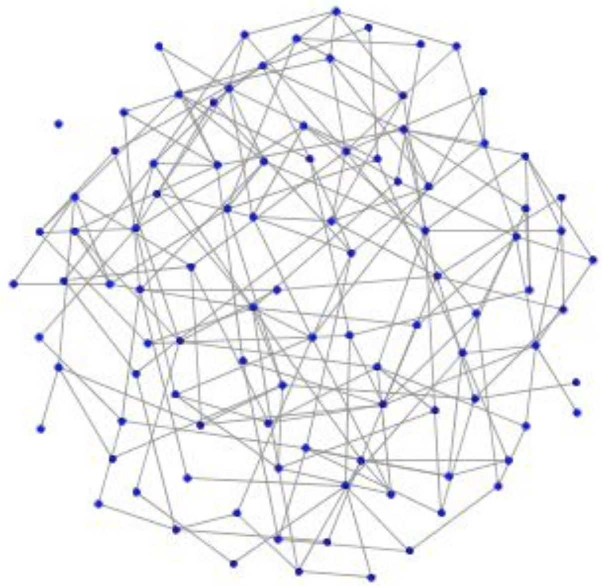
  - A and C are connected by 2 shortest paths
    - A – E – B - C
    - A – E – D - C

- Diameter: the largest geodesic distance in the graph
  - The distance between A and C is the maximum for the graph: 3

- Caution: some people use the term 'diameter' to be the average shortest path distance, in this class we will use it only to refer to the maximal distance

Characterizing networks:
How dense are they?

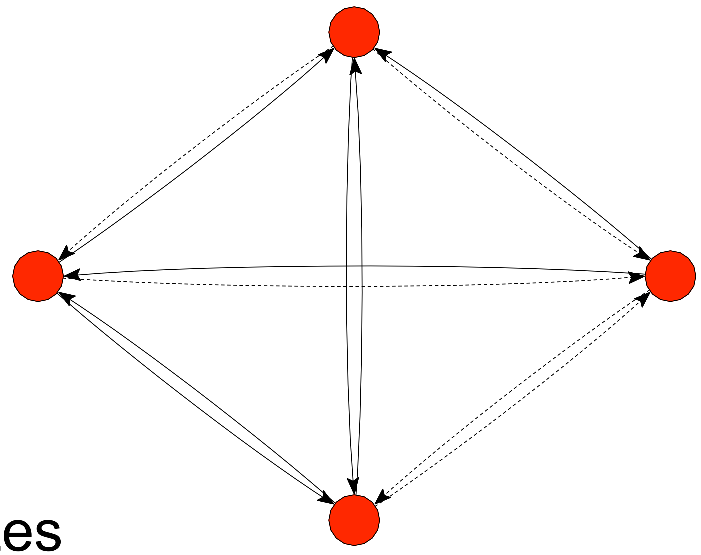




network metrics: graph density

- Of the connections that may exist between n nodes
  - directed graph
    $e_{max} = n*(n-1)$
    each of the n nodes can connect to (n-1) other nodes
  - undirected graph
    $e_{max} = n*(n-1)/2$
    since edges are undirected, count each one only once

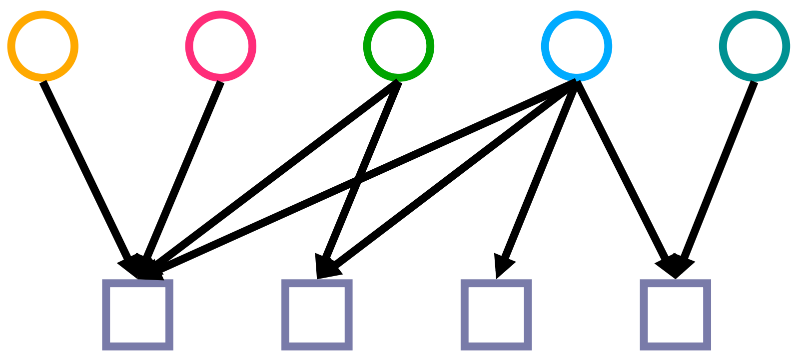
- What fraction are present?
  - density = e/ e_{max}
  - For example, out of 12 possible connections, this graph has 7, giving it a density of 7/12 = 0.583



- Would this measure be useful for comparing networks of different sizes (different numbers of nodes)?

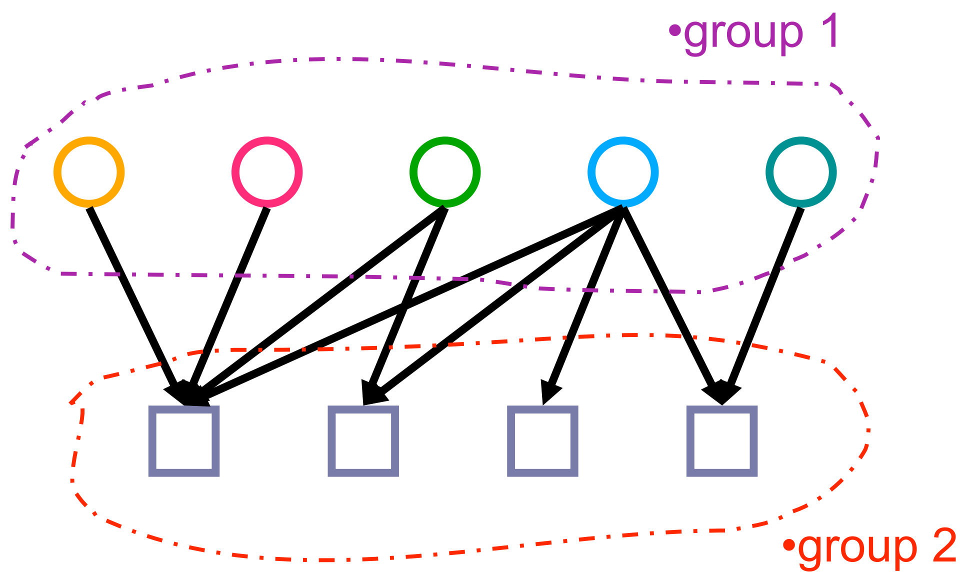
bipartite (two-mode) networks

- edges occur only between two groups of nodes, not within those groups
- for example, we may have individuals and *events*
  - directors and boards of directors
  - customers and the items they purchase
  - metabolites and the reactions they participate in

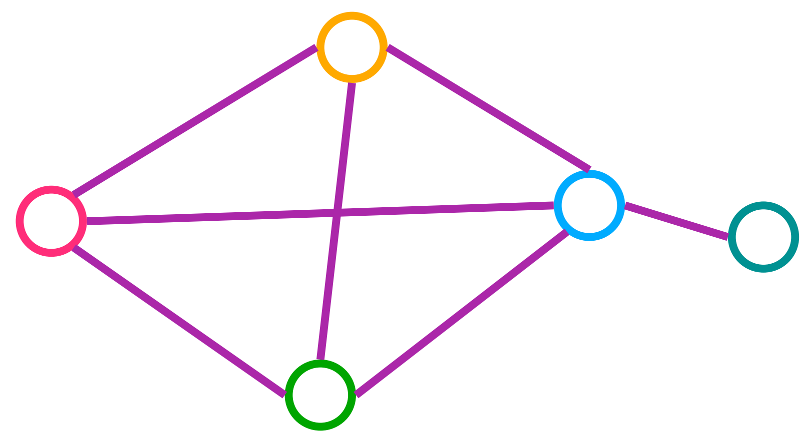


going from a bipartite to a one-mode graph

- Two-mode network



- One mode projection
  - two nodes from the first group are connected if they link to the same node in the second group
  - some loss of information
  - naturally high occurrence of cliques

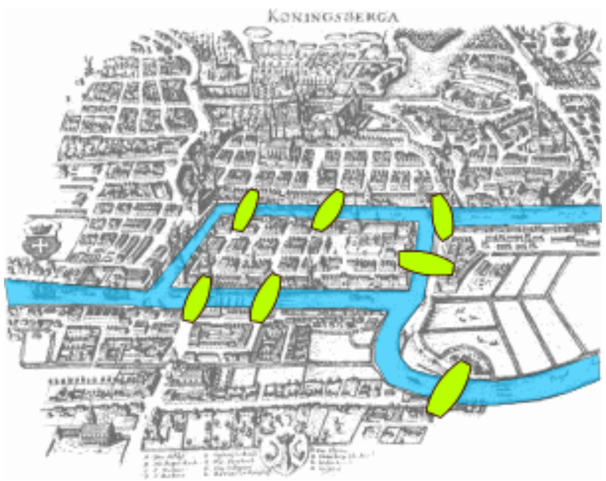


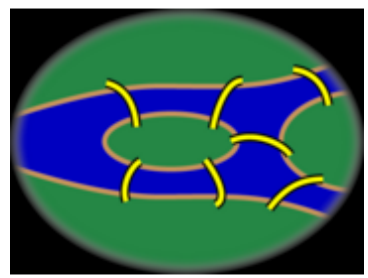
Outline

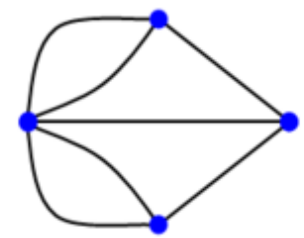
- What is a network?
  - a bunch of nodes and edges
- How do you characterize it?
  - with some basic network metrics
- How did network analysis get started
  - it was the mathematicians
- How do you analyze networks today?
  - with pajek or other software

History: Graph theory

- Euler’s **Seven Bridges of Königsberg** – one of the first problems in graph theory
- Is there a route that crosses each bridge only once and returns to the starting point?



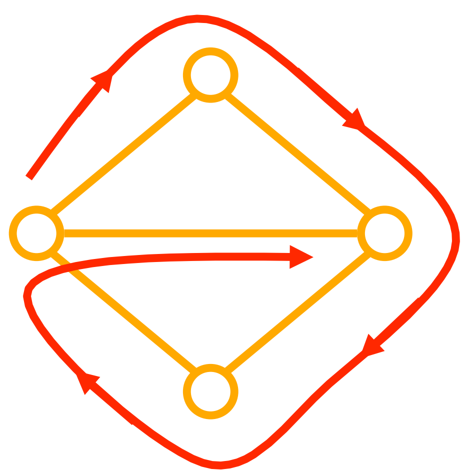




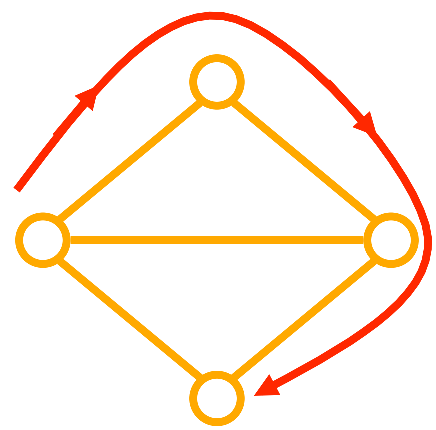
Source: http://en.wikipedia.org/wiki/Seven_Bridges_of_Königsberg

Eulerian paths

- If starting point and end point are the same:
  - only possible if no nodes have an odd degree
    - each path must visit and leave each shore
- If don’t need to return to starting point
  - can have 0 or 2 nodes with an odd degree



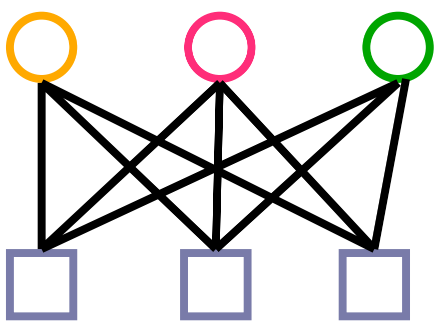
•Eulerian path: traverse each
•edge exactly once



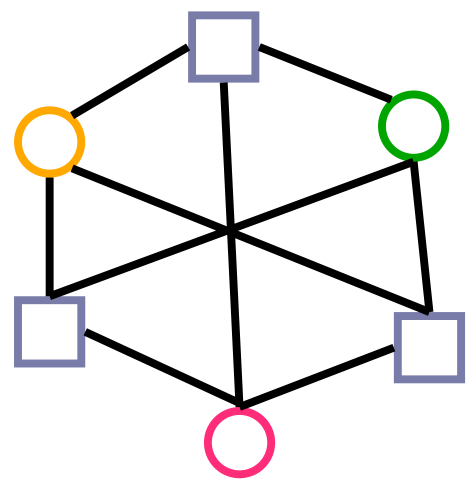
•Hamiltonian path: visit
•each vertex exactly once

Bi-cliques (cliques in bipartite graphs)

- $K_{m,n}$ is the complete bipartite graph with **m** and **n** vertices of the two different types
- $K_{3,3}$ maps to the utility graph
  - Is there a way to connect three utilities, e.g. gas, water, electricity to three houses without having any of the pipes cross?



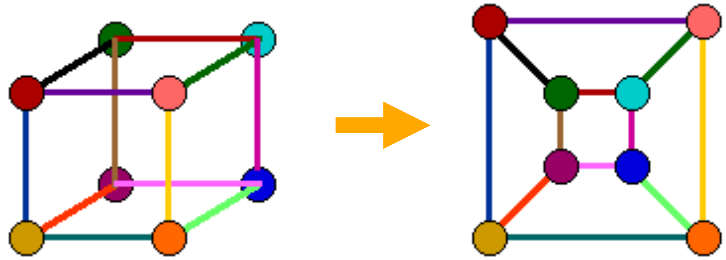
•$K_{3,3}$



•Utility graph

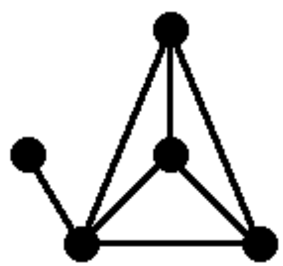
Planar graphs

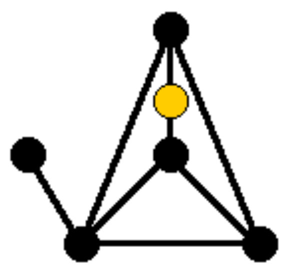
- A graph is planar if it can be drawn on a plane without any edges crossing



When graphs are not planar

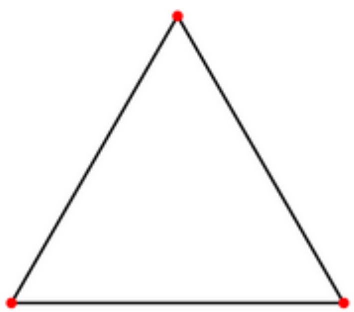
- Two graphs are **homeomorphic** if you can make one into the other by adding a vertex of degree 2



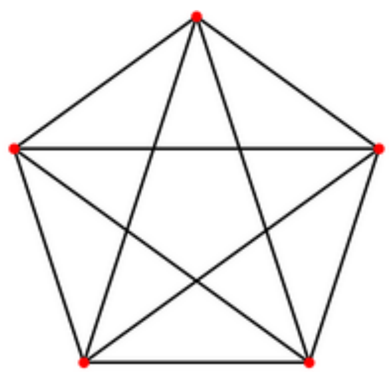


Cliques and complete graphs

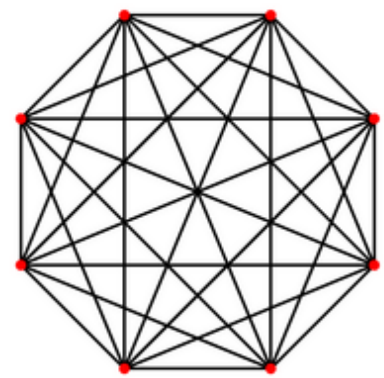
- K_n is the complete graph (clique) with K vertices
  - each vertex is connected to every other vertex
  - there are n*(n-1)/2 undirected edges



•K_3



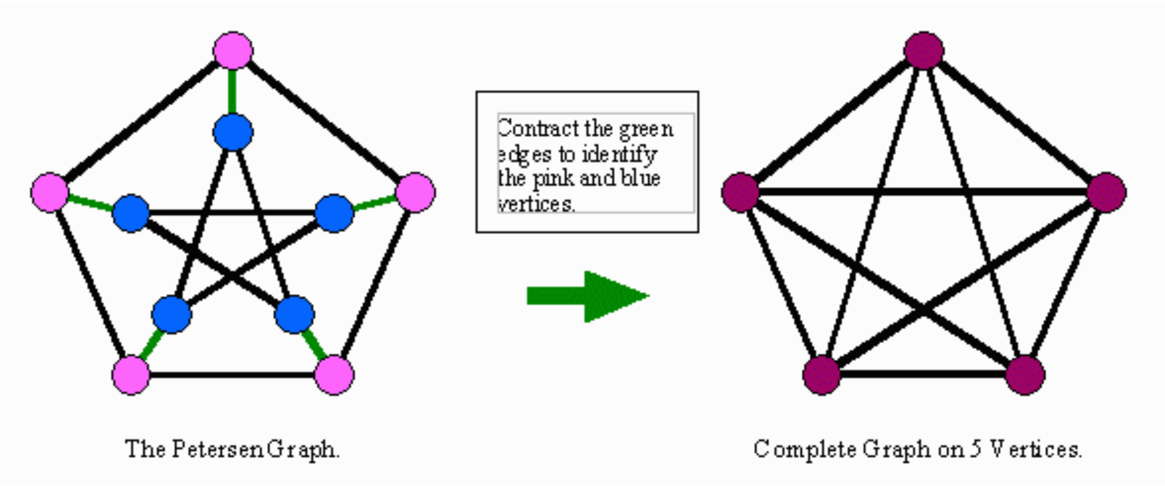
•K_5



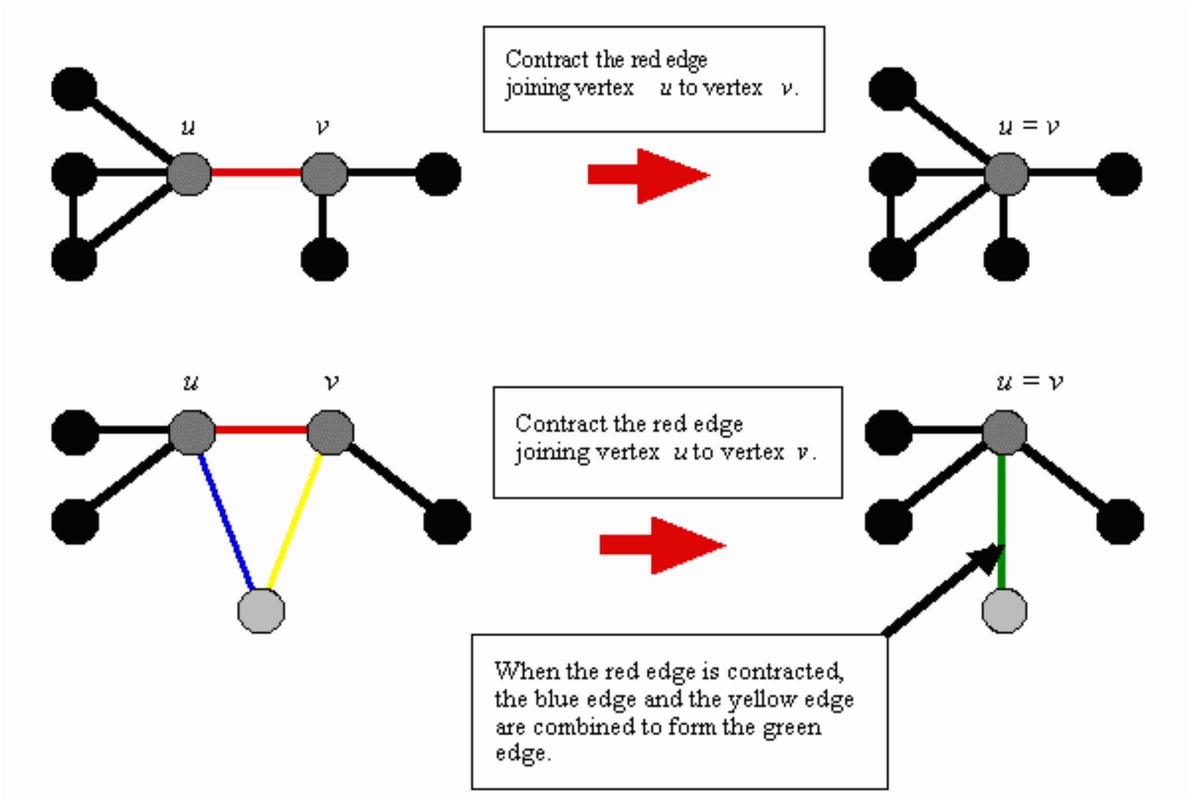
•K_8

Peterson graph

- Example of using edge contractions to show a graph is not planar



Edge contractions defined



- A finite graph G is planar if and only if it has no subgraph that is homeomorphic or edge-contractible to the complete graph in five vertices (K_5) or the complete bipartite graph $K_{3,\,3}$. **(Kuratowski's Theorem)**

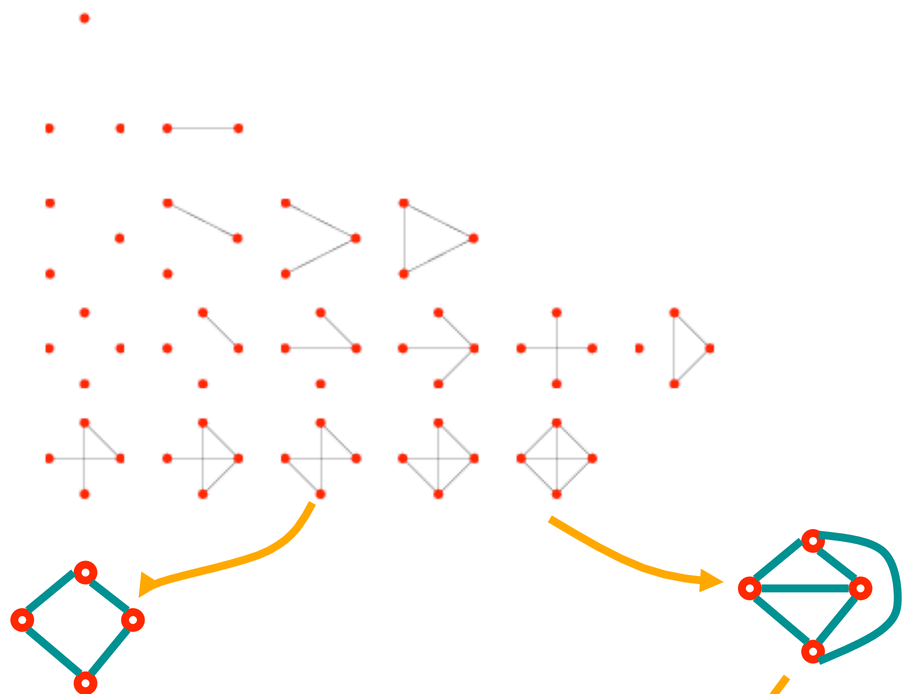
#s of planar graphs of different sizes

•1:1

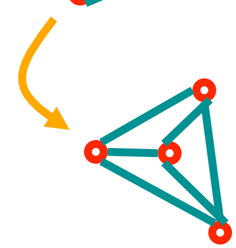
•2:2

•3:4

•4:11

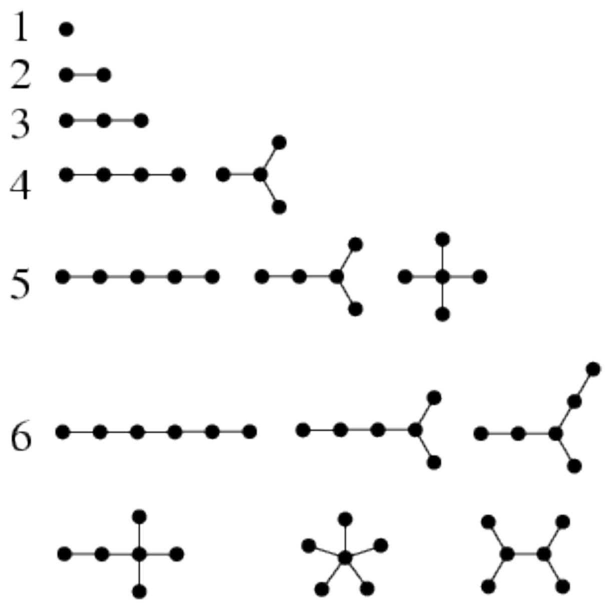


•Every planar graph
•has a straight line
•embedding



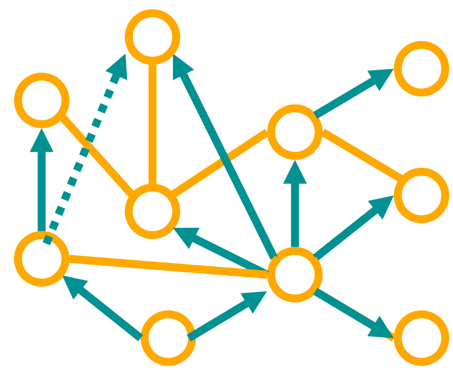
Trees

- Trees are undirected graphs that contain no cycles



examples of trees

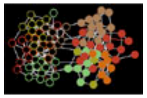
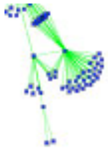
- In nature
- Man made
- Computer science
- Network analysis



Outline

- What is a network?
  - a bunch of nodes and edges
- How do you characterize it?
  - with some basic network metrics
- How did network analysis get started?
  - it was the mathematicians
- How do you analyze networks today?
  - with pajek or other software

overview of network analysis tools

	Pajek	network analysis and visualization, menu driven, suitable for large networks	platforms: Windows (on linux via Wine) download
	**Netlogo**	agent based modeling recently added network modeling capabilities	platforms: any (Java) download
	**GUESS**	network analysis and visualization, extensible, script-driven (jython)	platforms: any (Java) download

Other software tools that we will not be using but that you may find useful:

visualization and analysis:

UCInet - user friendly social network visualization and analysis software (suitable smaller networks)
iGraph - if you are familiar with R, you can use iGraph as a module to analyze or create large networks, or you can directly use the C functions
Jung - comprehensive Java library of network analysis, creation and visualization routines
Graph package for Matlab (untested?) - if Matlab is the environment you are most comfortable in, here are some basic routines
SIENA - for p* models and longitudinal analysis
SNA package for R - all sorts of analysis + heavy duty stats to boot
NetworkX - python based free package for analysis of large graphs
InfoVis Cyberinfrastructure - large agglomeration of network analysis tools/routines, partly menu driven

visualization only:

GraphViz - open source network visualization software (can handle large/specialized networks)
TouchGraph - need to quickly create an interactive visualization for the web?
yEd - free, graph visualization and *editing* software

specialized:

fast community finding algorithm
motif profiles
CLAIR library - NLP and IR library (Perl Based) includes network analysis routines

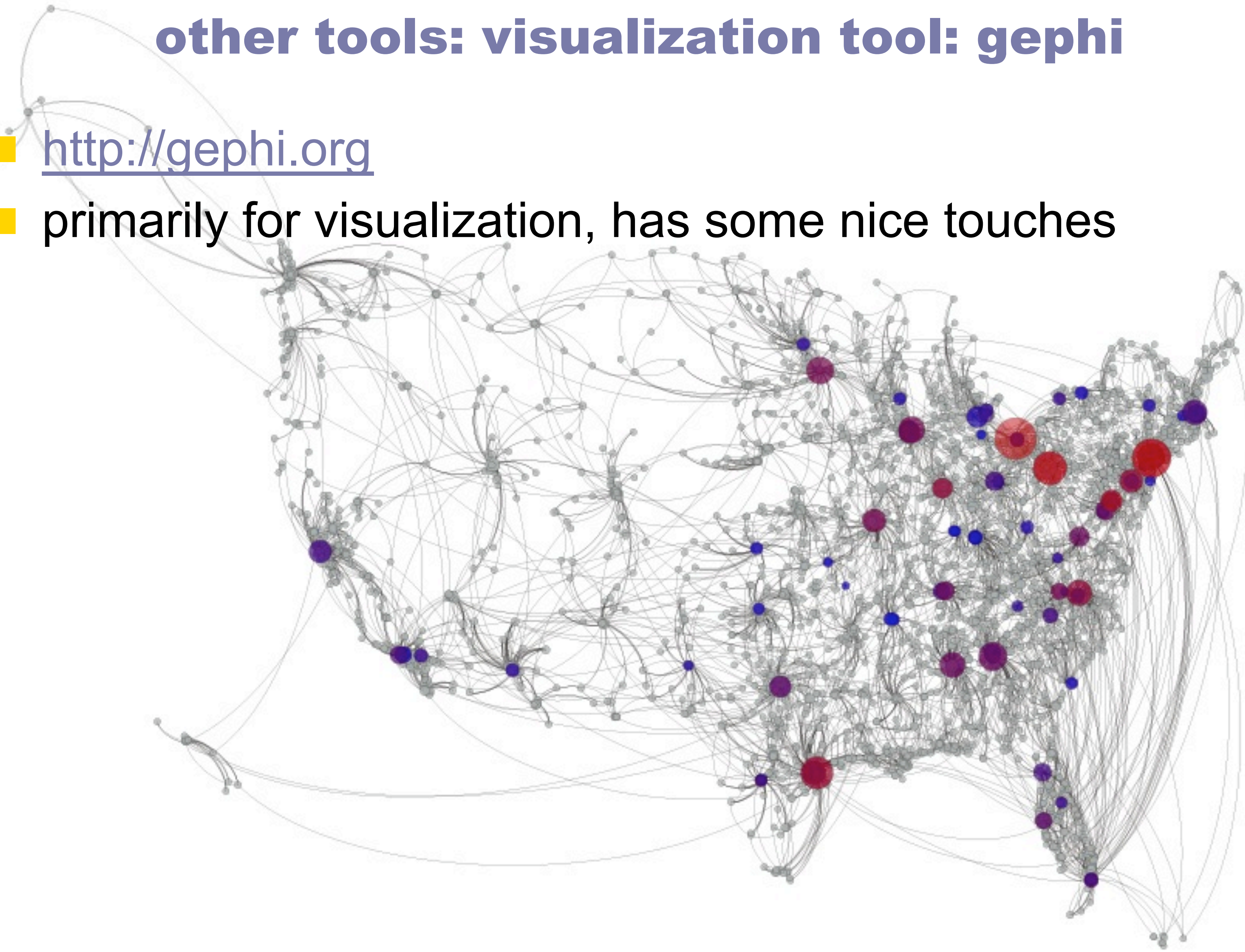
finally: INSNA long list of SNA packages

tools we'll use

- **Pajek:** extensive menu-driven functionality, including many, many network metrics and manipulations
  - but… not extensible
- **Guess:** extensible, scriptable tool of exploratory data analysis, but more limited selection of built-in methods compared to Pajek
- **NetLogo:** general agent based simulation platform with excellent network modeling support
  - many of the demos in this course were built with NetLogo
- **iGraph:** used in PhD-level version of this course. libraries can be accessed through R or python. Routines scale to millions of nodes.

other tools: visualization tool: gephi

- http://gephi.org
- primarily for visualization, has some nice touches



visualization tool: GraphViz

- Takes descriptions of graphs in simple text languages
- Outputs images in useful formats
- Options for shapes and colors
- Standalone or use as a library

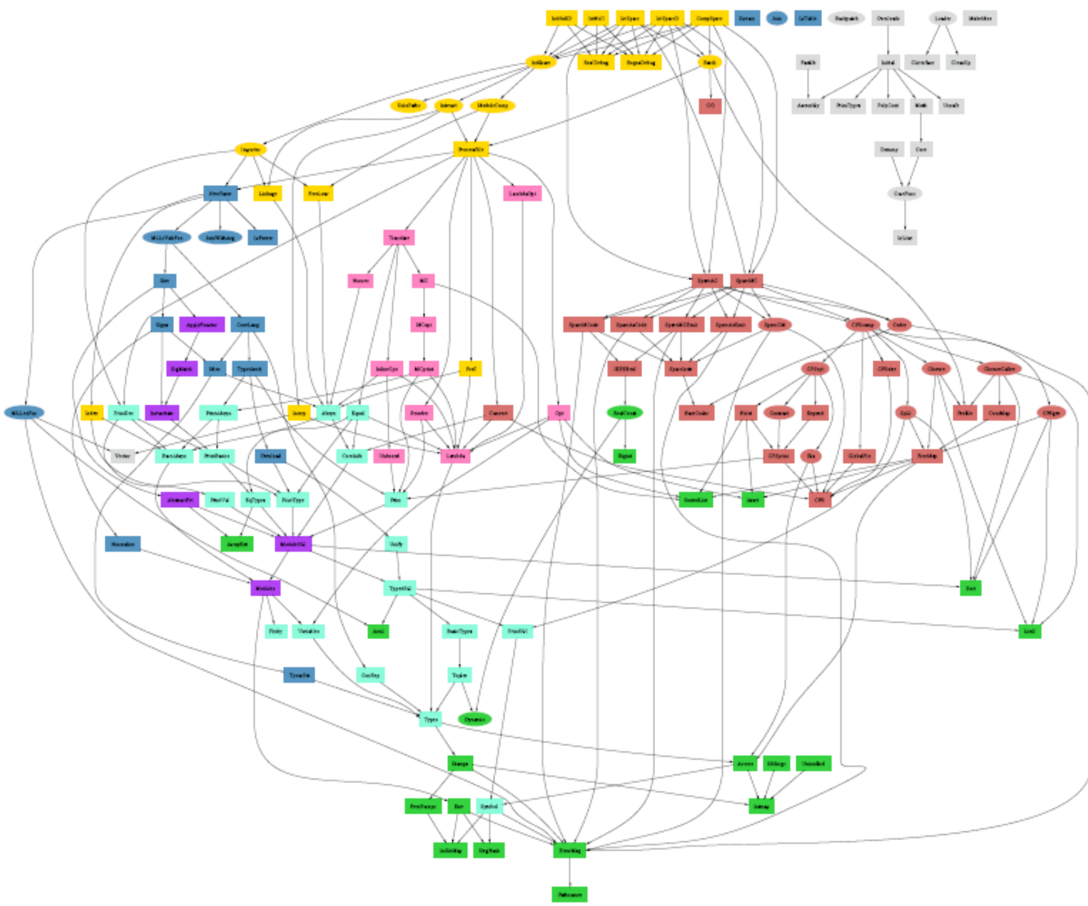
- dot: hierarchical or layered drawings of directed graphs, by avoiding edge crossings and reducing edge length
- neato (Kamada-Kawai) and fdp (Fruchterman-Reinhold with heuristics to handle larger graphs)
- twopi – radial layout
- circo – circular layout

http://www.graphviz.org/

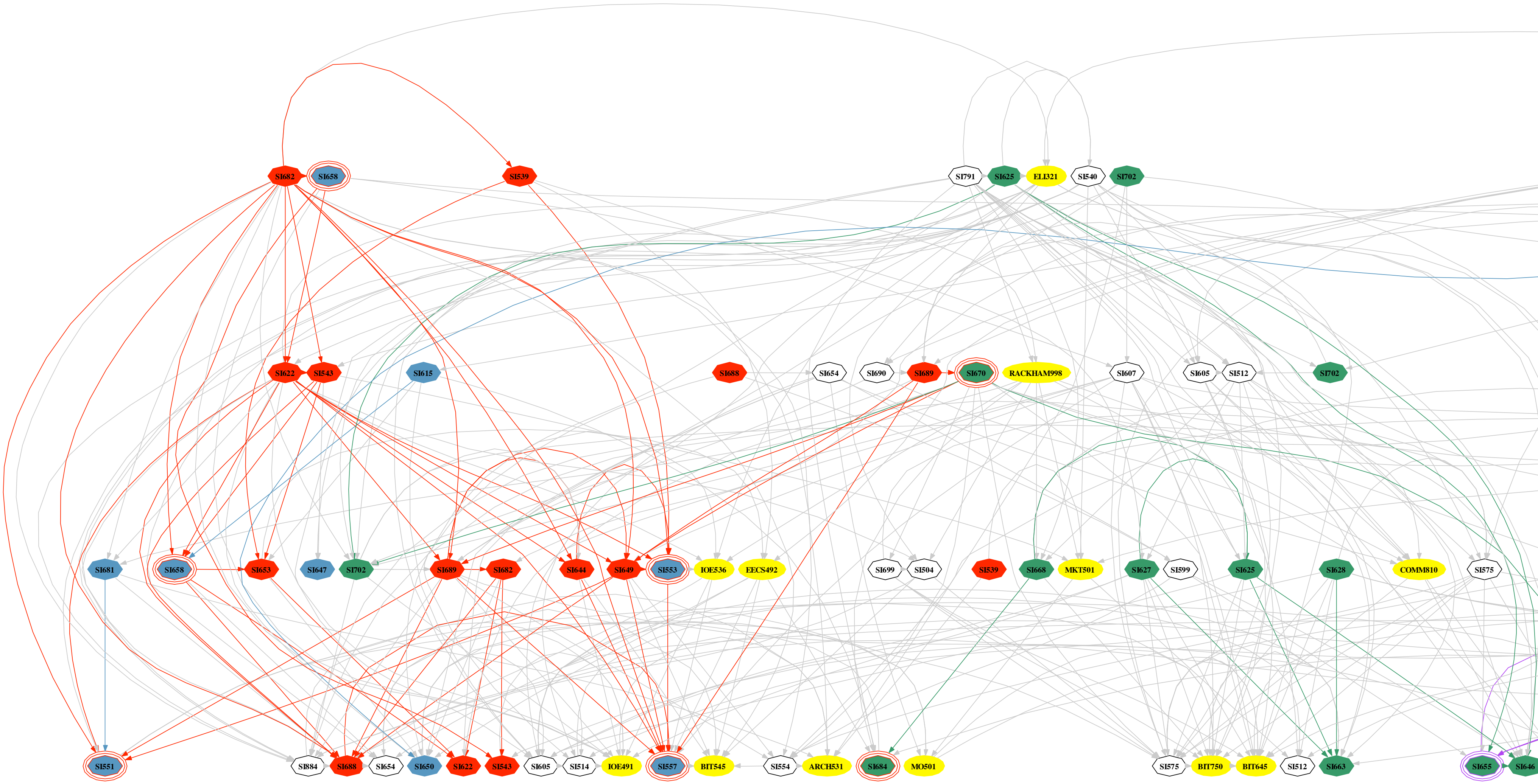
GraphViz: dot language

```
digraph G {
  ranksep=4
  nodesep=0.1
  size="8,11"
ARCH531_20061 [label="ARCH531",style=bold,color=yellow,style=filled]
ARCH531_20071 [label="ARCH531",gstyle=bold,color=yellow,style=filled]
BIT512_20071 [label="BIT512",gstyle=bold,color=yellow,style=filled]
BIT513_20071 [label="BIT513",gstyle=bold,color=yellow,style=filled]
BIT646_20064 [label="BIT646",gstyle=bold,color=yellow,style=filled]
BIT648_20064 [label="BIT648",gstyle=bold,color=yellow,style=filled]
DESCI502_20071 [label="DESCI502",gstyle=bold,color=yellow,style=filled]
ECON500_20064 [label="ECON500",gstyle=bold,color=yellow,style=filled]
…
…
SI791_20064->SI549_20064[weight=2,color=slategray,style="setlinewidth(4)"]SI791_20064-
    >SI596_20071[weight=5,color=slategray,style=bold,style="setlinewidth(10)"]SI791_20064-
    >SI616_20071[weight=2,color=slategray,style=bold,style="setlinewidth(4)"]SI791_20064-
    >SI702_20071[weight=2,color=slategray,style=bold,style="setlinewidth(4)"]SI791_20064-
    >SI719_20071[weight=2,color=slategray,style=bold,style="setlinewidth(4)"]
```

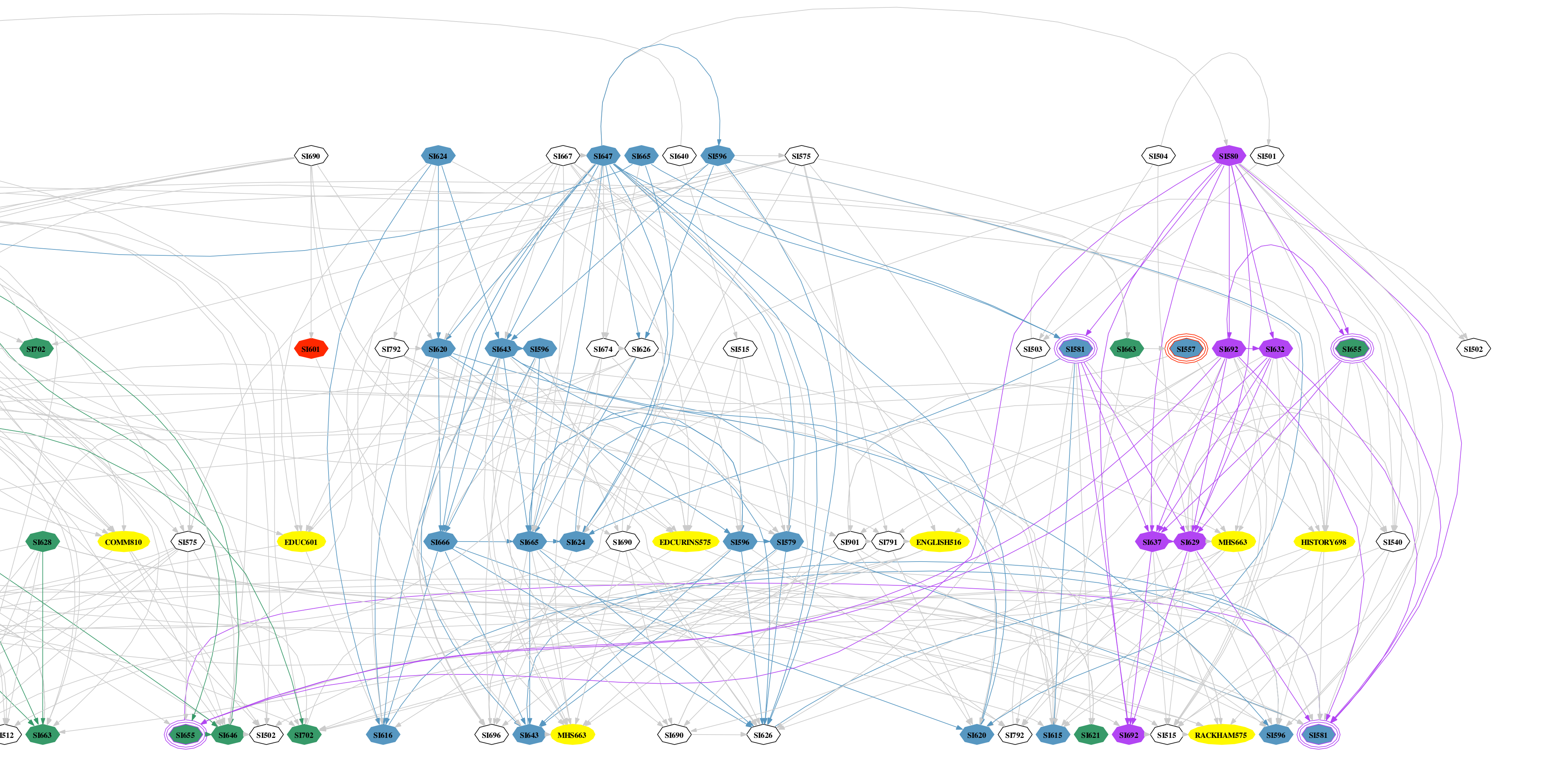
Dot (GraphViz)



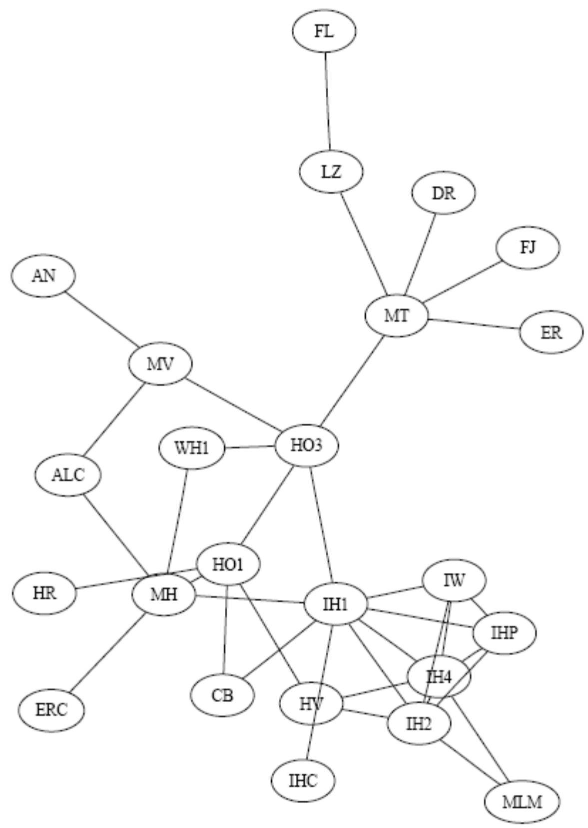
Lada's school of information course recommender (GraphViz)



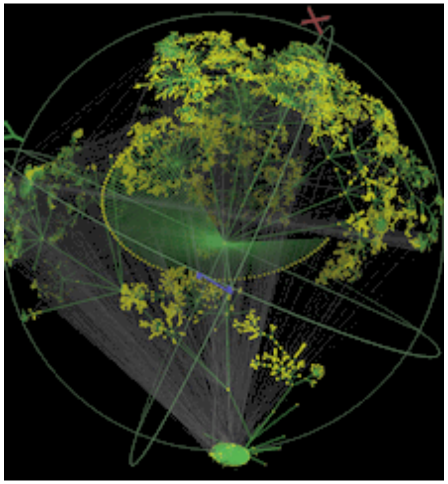
Lada's school of information course recommender (GraphViz)



Neato (Graphviz)



Other visualization tools: Walrus



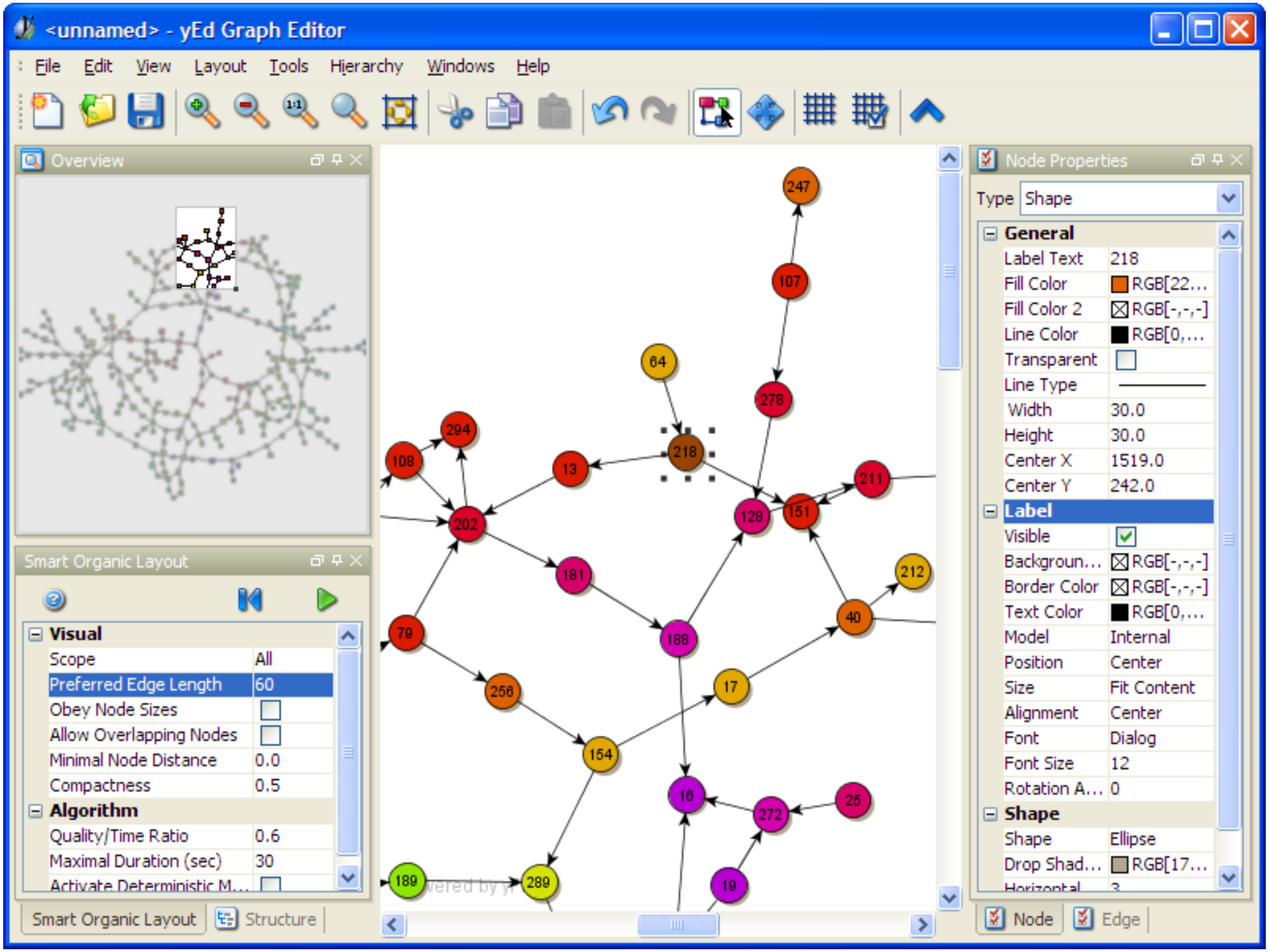
- developed at CAIDA available under the GNU GPL.
- "…best suited to visualizing moderately sized graphs that are nearly trees. A graph with a few hundred thousand nodes and only a slightly greater number of links is likely to be comfortable to work with."
- Java-based
- Implemented Features
  - rendering at a guaranteed frame rate regardless of graph size
  - coloring nodes and links with a fixed color, or by RGB values stored in attributes
  - labeling nodes
  - picking nodes to examine attribute values
  - displaying a subset of nodes or links based on a user-supplied boolean attribute
  - interactive pruning of the graph to temporarily reduce clutter and occlusion
  - zooming in and out

Source: CAIDA, http://www.caida.org/tools/visualization/walrus/

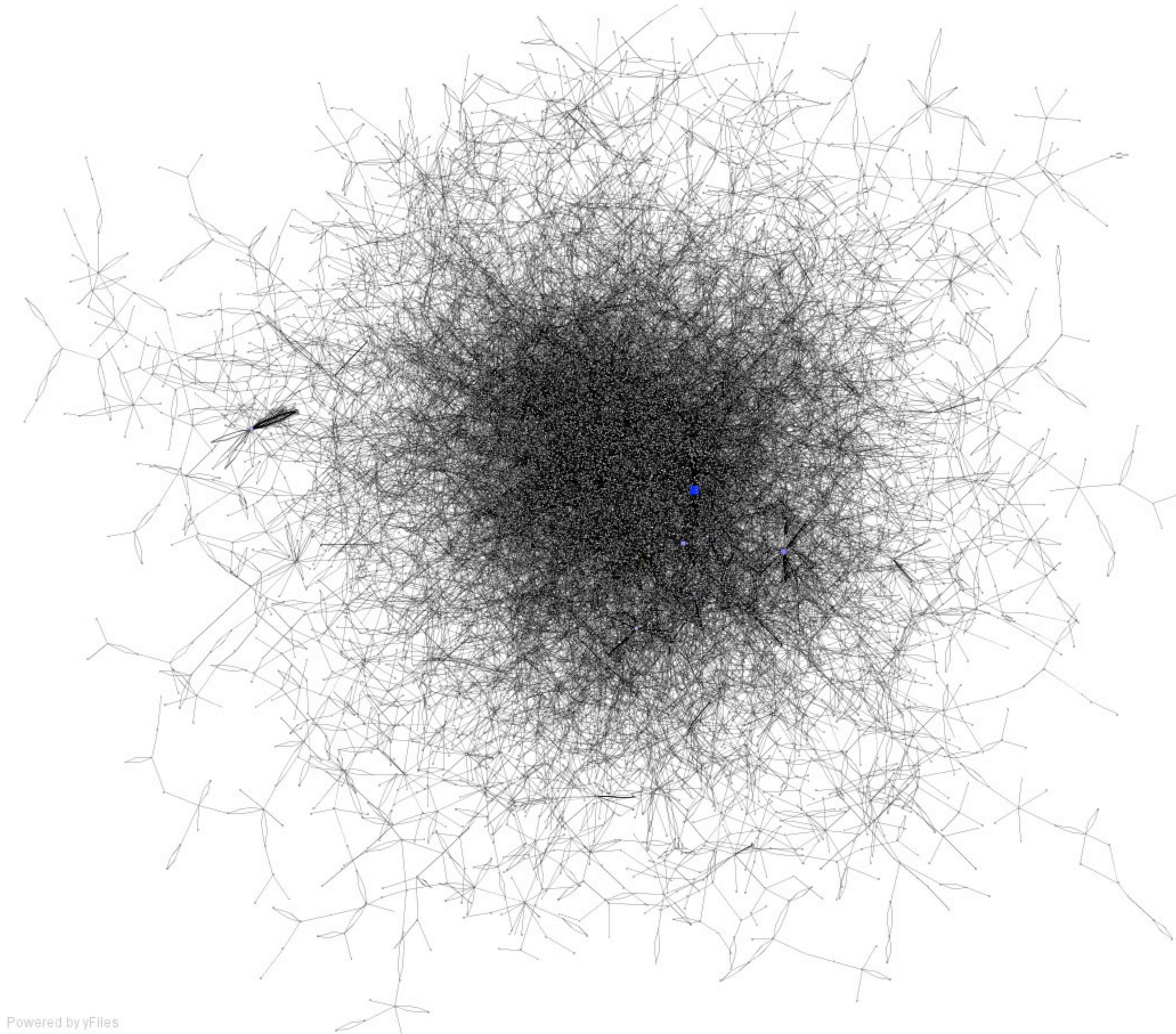
visualization tools: YEd - JavaTM Graph Editor

http://www.yworks.com/en/products_yed_about.htm

(good primarily for layouts, maybe free)



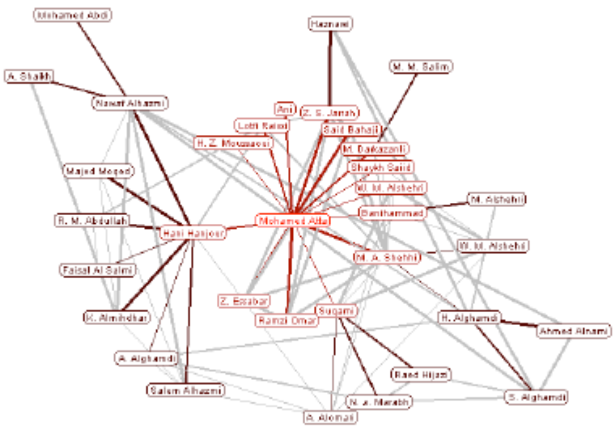
yEd and 26,000 nodes (takes a few seconds)



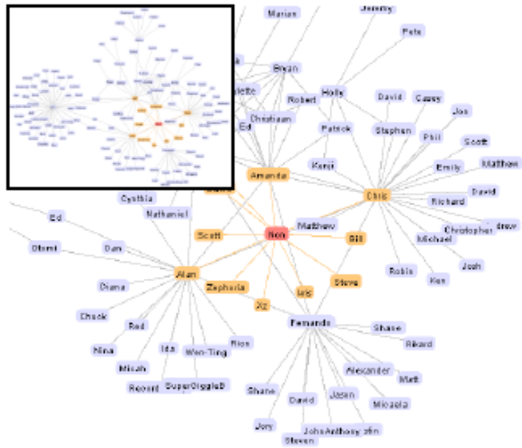
visualization tools: Prefuse

- (free) user interface toolkit for interactive information visualization
  - built in Java using Java2D graphics library
  - data structures and algorithms
  - pipeline architecture featuring reusable, composable modules
  - animation and rendering support
  - architectural techniques for scalability
- requires knowledge of Java programming
- website: http://prefuse.sourceforge.net/
  - CHI paper http://guir.berkeley.edu/pubs/chi2005/**prefuse**.pdf

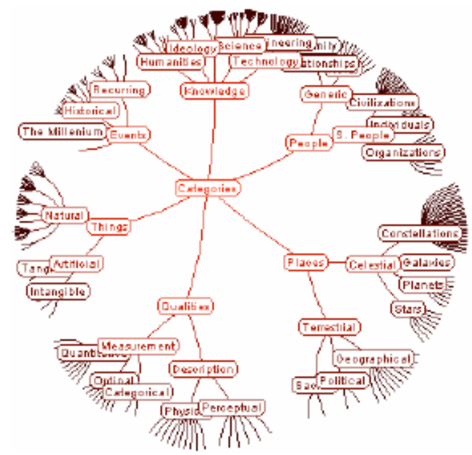
Simple prefuse visualizations



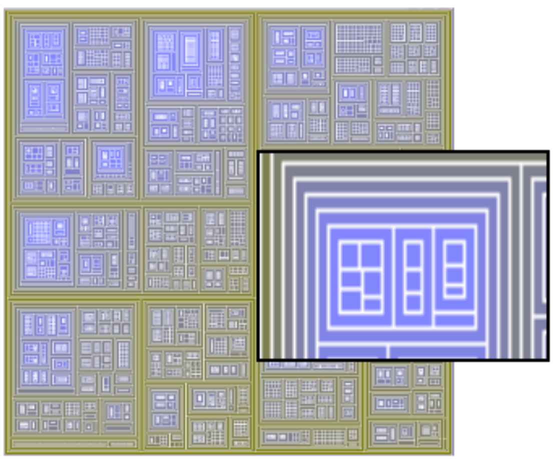
(a) Animated radial layout.



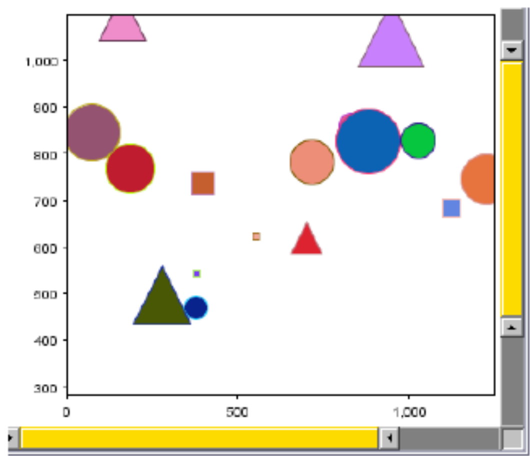
(b) Force-directed layout with overview.



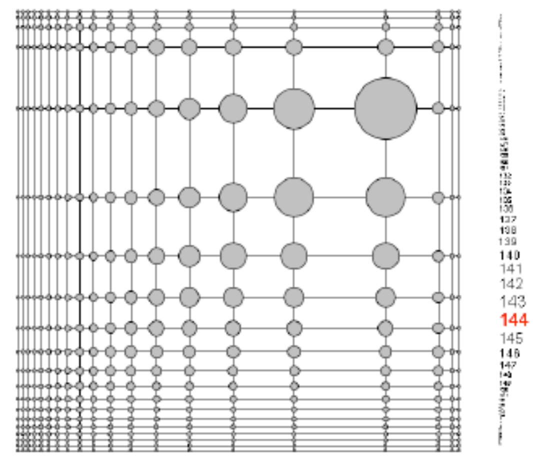
(c) Hyperbolic tree.



(d) TreeMap.



(e) SpotPlot scatterplot.

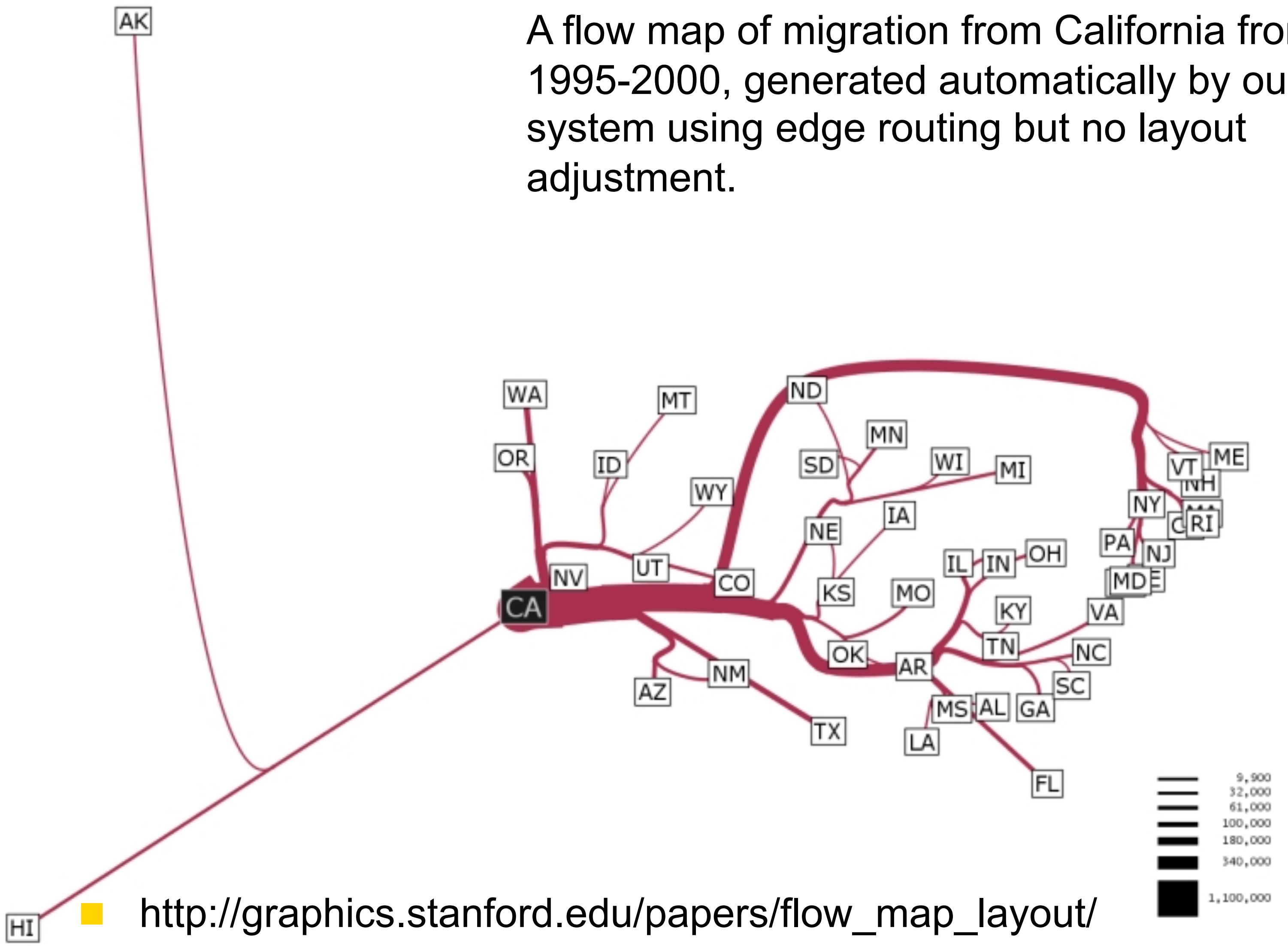


(f) Fisheye graph. **(g)** Fisheye menu.

Source: Prefuse, http://prefuse.sourceforge.net/

Examples of prefuse applications: flow maps

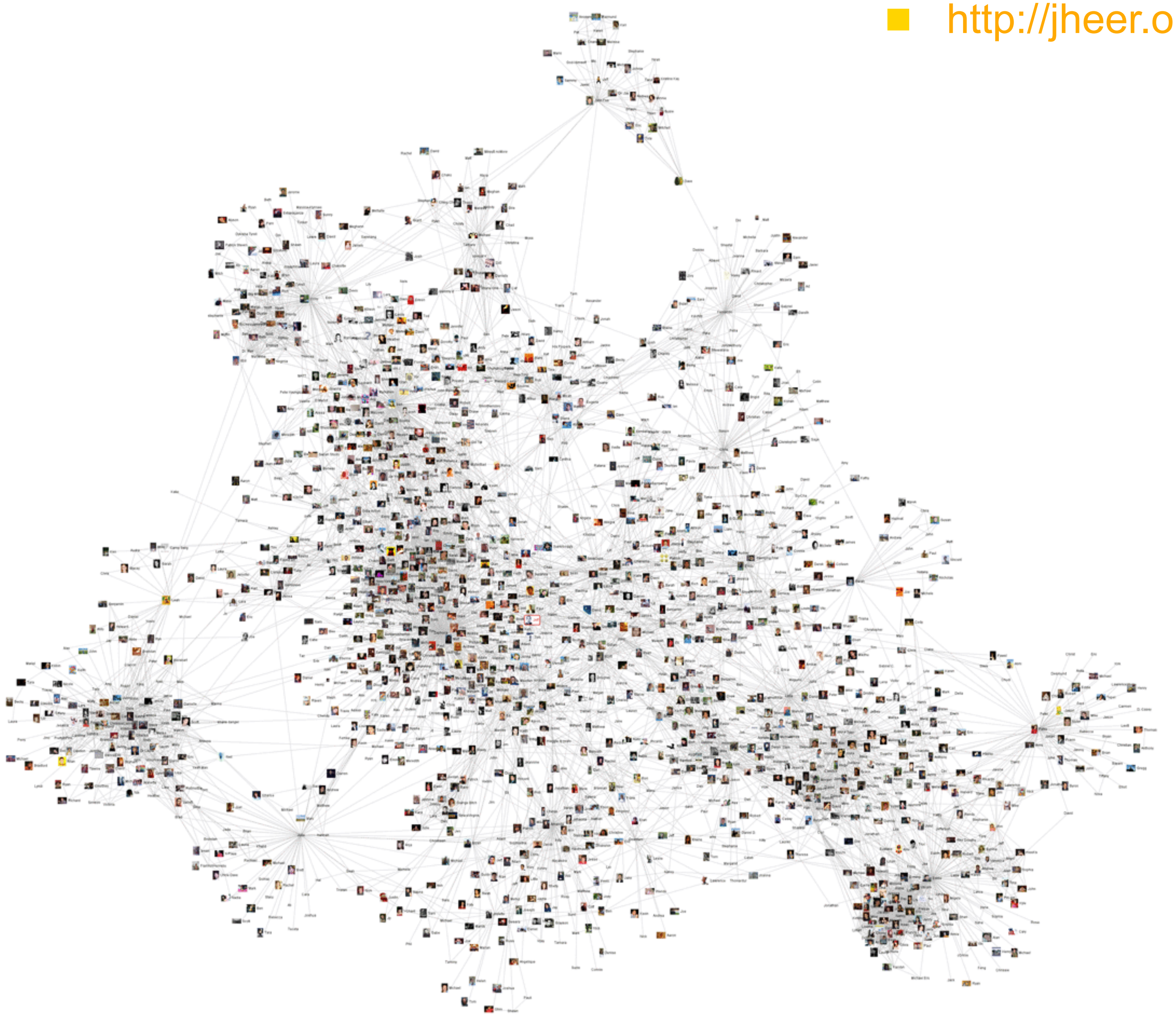
A flow map of migration from California from 1995-2000, generated automatically by our system using edge routing but no layout adjustment.



- http://graphics.stanford.edu/papers/flow_map_layout/

Examples of prefuse applications: vizster

- http://jheer.org/vizster/



Outline

- Network metrics can help us characterize networks
- This has is roots in graph theory
- Today there are many network analysis tools to choose from
  - though most of them are in beta!

- In lab: exploratory network analysis with Pajek