Network Analysis using Python

Muhammad Qasim Pasta

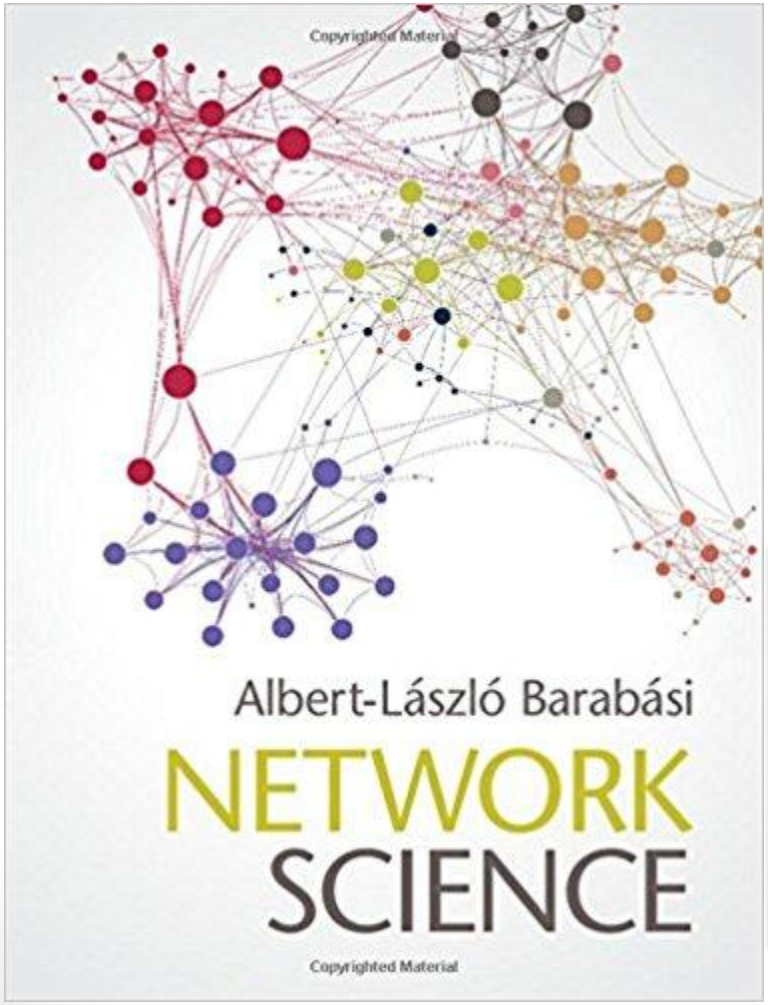
Assistant Professor
Usman Institute of Technology

About Me

- Assistant Professor @ UIT
- Research Interest: Network Science / Social Network Analysis
- Consultant / Mentor
- Python Evangelist

- Github: https://github.com/mqpasta
  - https://github.com/mqpasta/pyconpk2018
- Website: http://qasimpasta.info

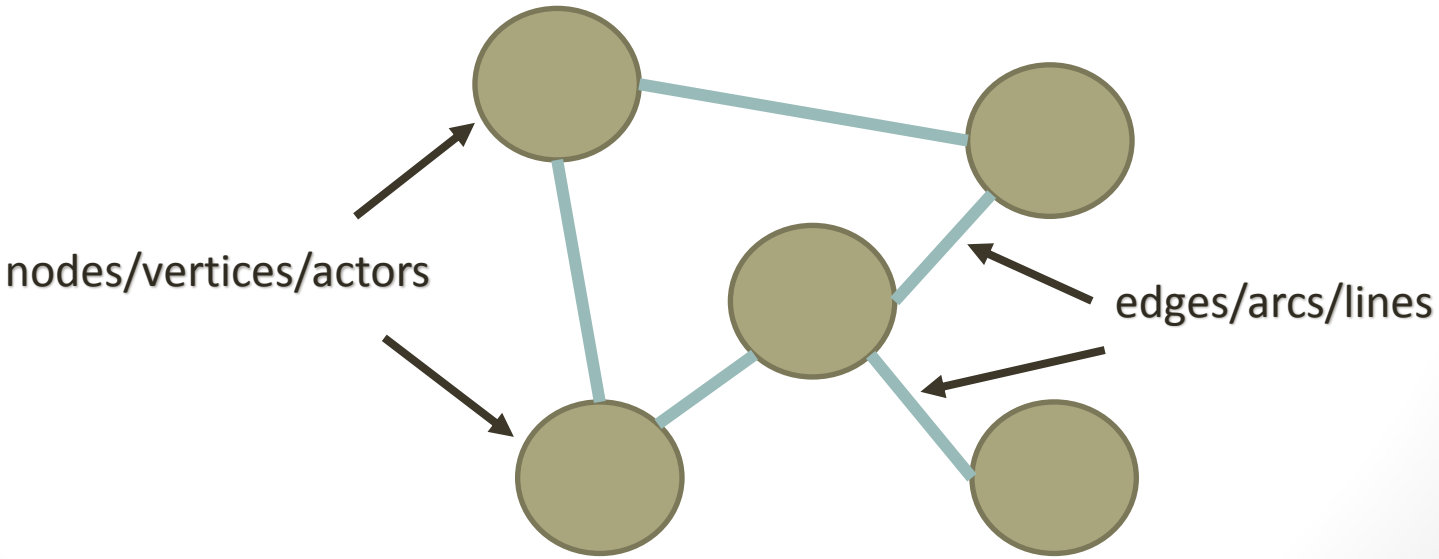
Today's Talk


Copyrighted Material
Albert-László Barabási
NETWORK
SCIENCE
Copyrighted Material


NetworkX

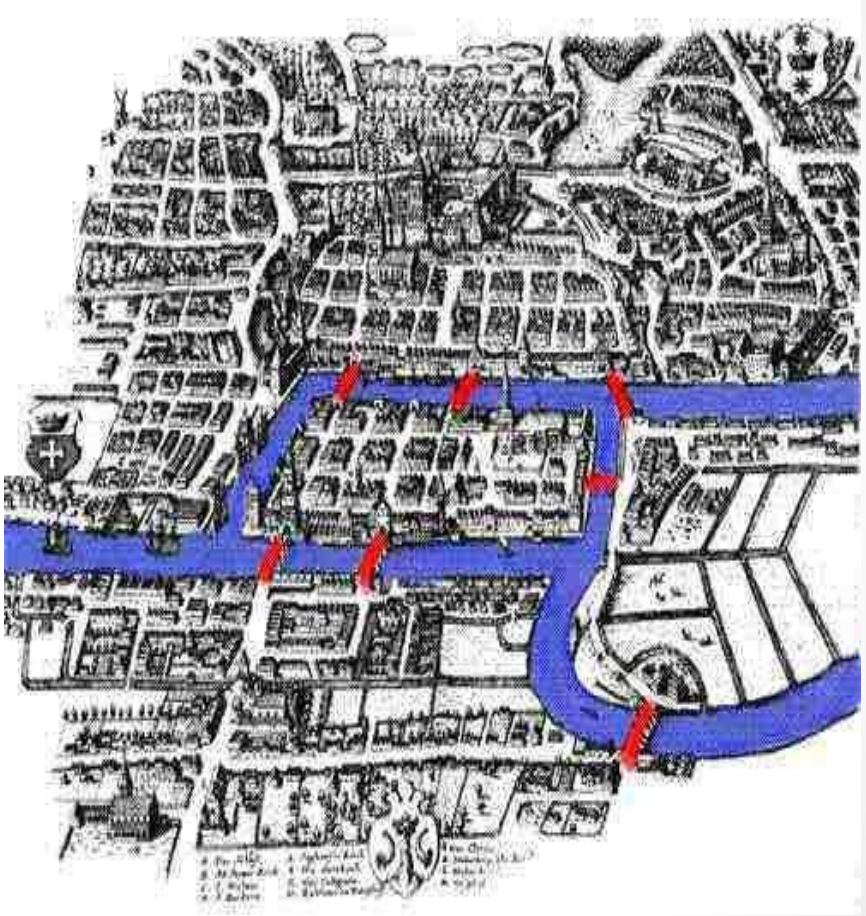
What is a Network?

- An abstract representation to model pairwise relationship between objects.
- Network = graph: mathematical representation



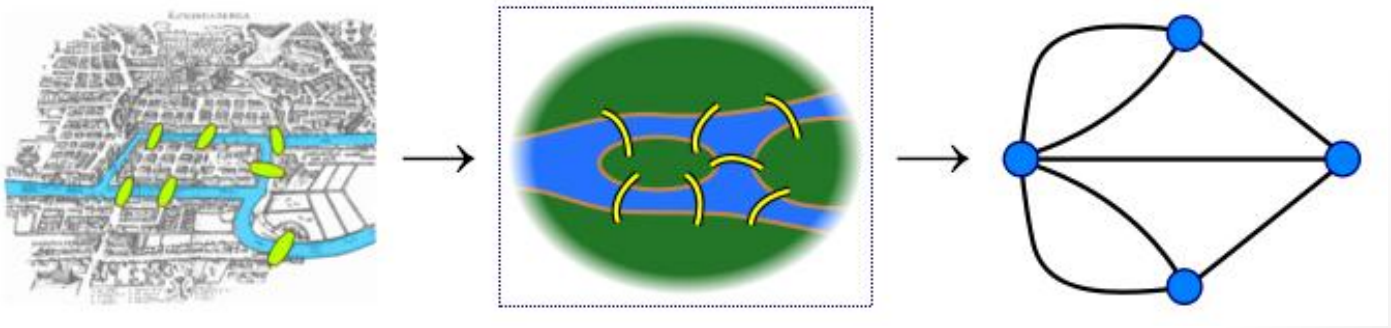
Birth of Graph Theory

- Königsberg bridge problem: the seven bridges of the city of **Königsberg** over the river Preger
- traversed in a single trip without doubling back?
- trip ends in the same place it began



Solution as graph!

- Swiss mathematician - Leonhard Euler
- considering each piece of island as dot
- each bridge connecting two island as a line between two dots
- a graph of dots (vertices or nodes) and lines (edges or links).

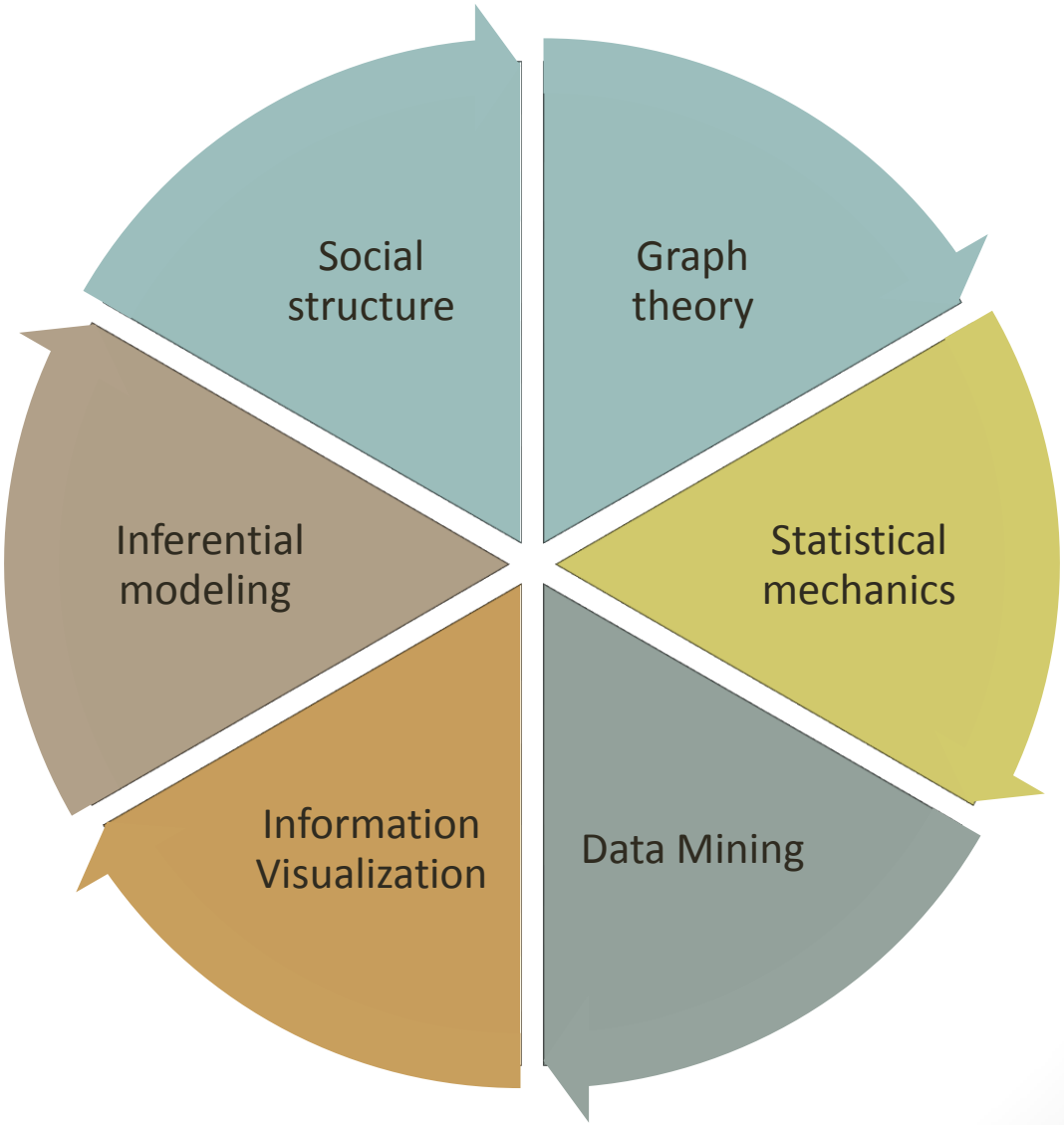


Network Science

- "**Network science** is an academic field which **studies complex networks** such as telecommunication networks, computer networks …" [Wikipedia]

- "In the context of network theory, a **complex network** is a graph (network) with **non-trivial topological features**—features that do not occur in simple networks …" [Wikipedia]

Study of Network Science

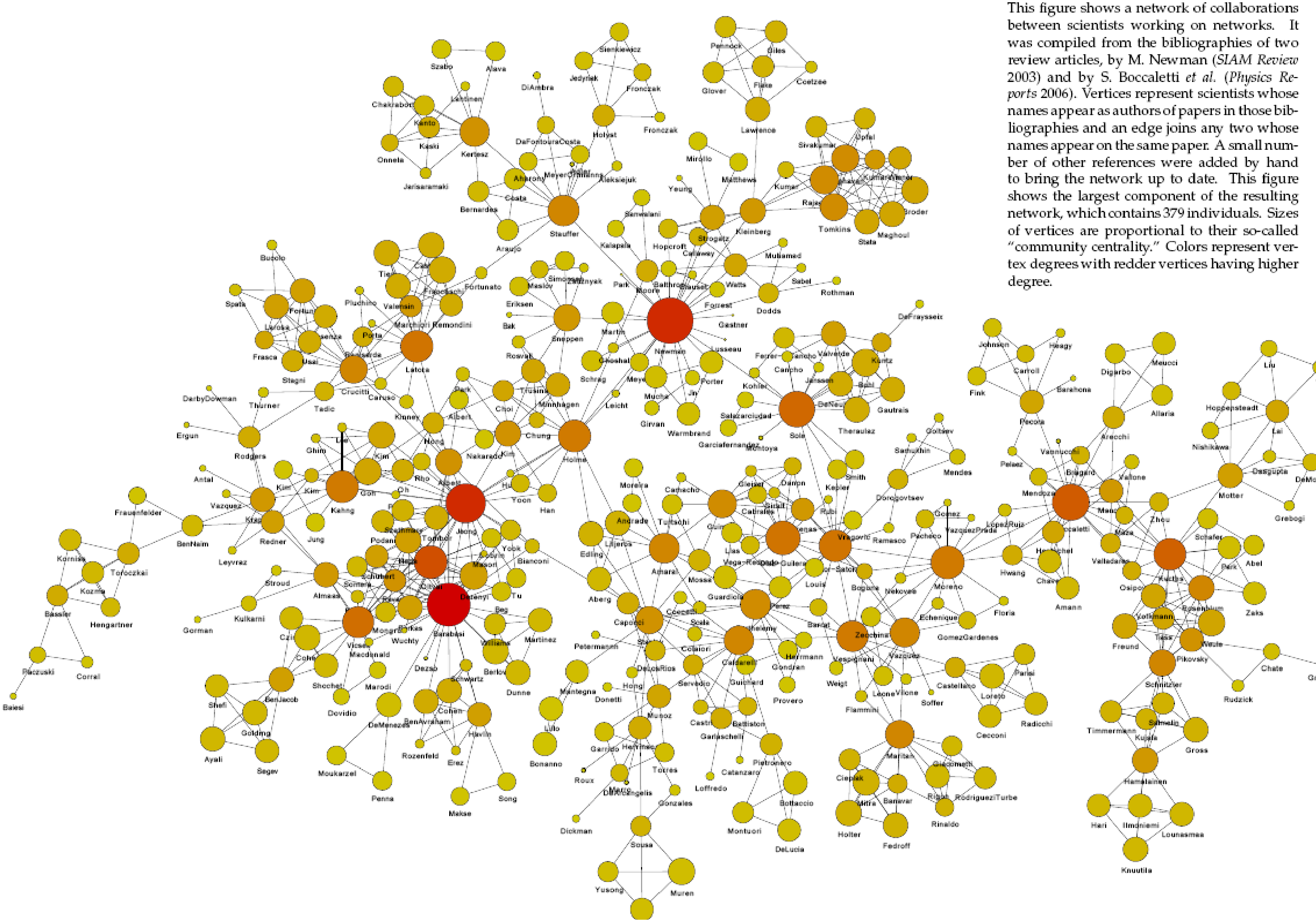


Why networks are important?

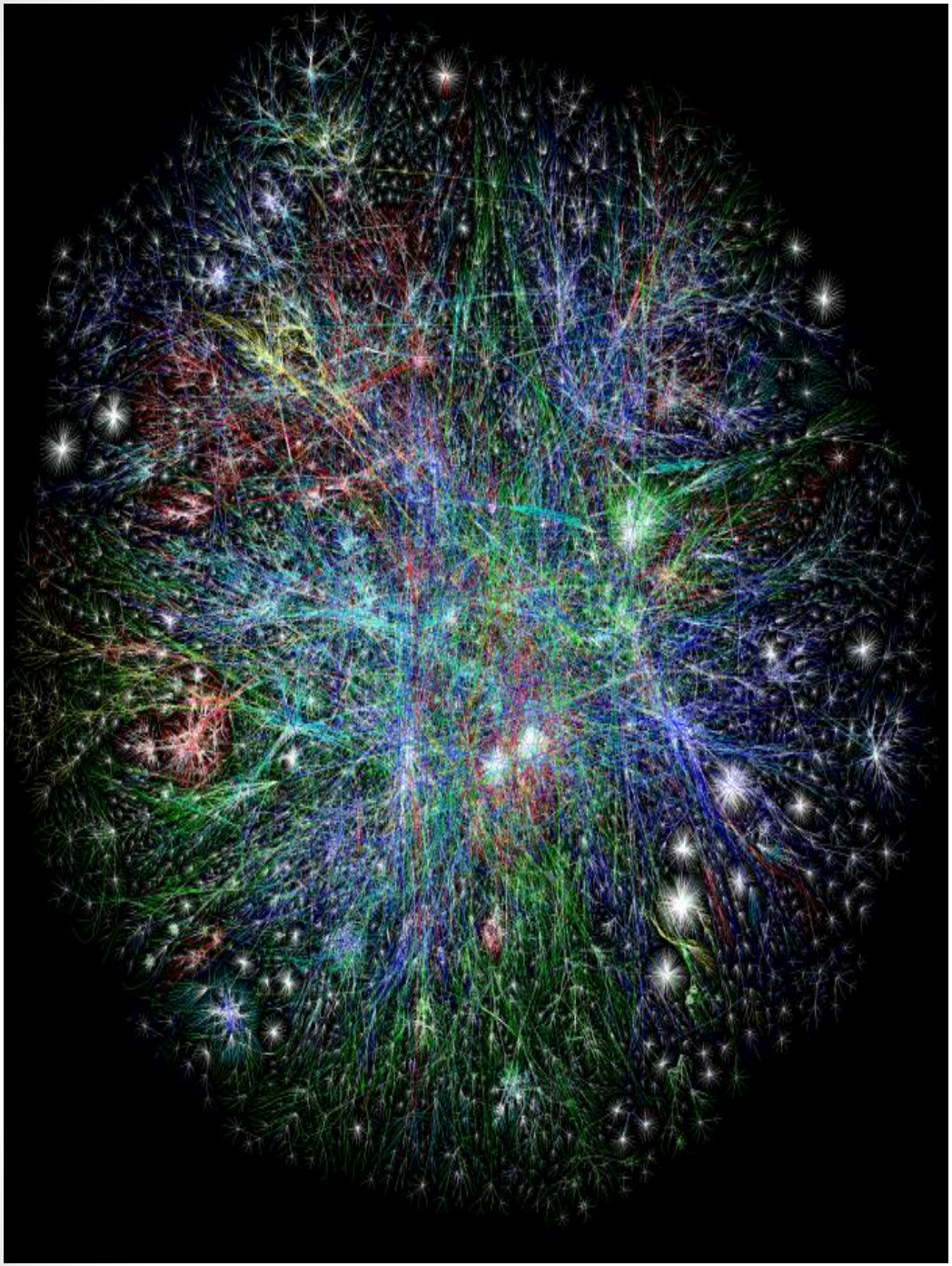
Duncan Watts (1971 -), Sociologist and principal researcher at Microsoft Research, New York City. He is known for his work on small-world networks

"Networks are important because if we don't understand networks, we can't understand how markets function, organizations solve problems, or how societies change" – Duncan Watts (Networks Science, 1, p.2)

Networks are everywhere! We are surrounded in the networks



This figure shows a network of collaborations between scientists working on networks. It was compiled from the bibliographies of two review articles, by M. Newman (*SIAM Review* 2003) and by S. Boccaletti *et al.* (*Physics Reports* 2006). Vertices represent scientists whose names appear as authors of papers in those bibliographies and an edge joins any two whose names appear on the same paper. A small number of other references were added by hand to bring the network up to date. This figure shows the largest component of the resulting network, which contains 379 individuals. Sizes of vertices are proportional to their so-called "community centrality." Colors represent vertex degrees with redder vertices having higher degree.

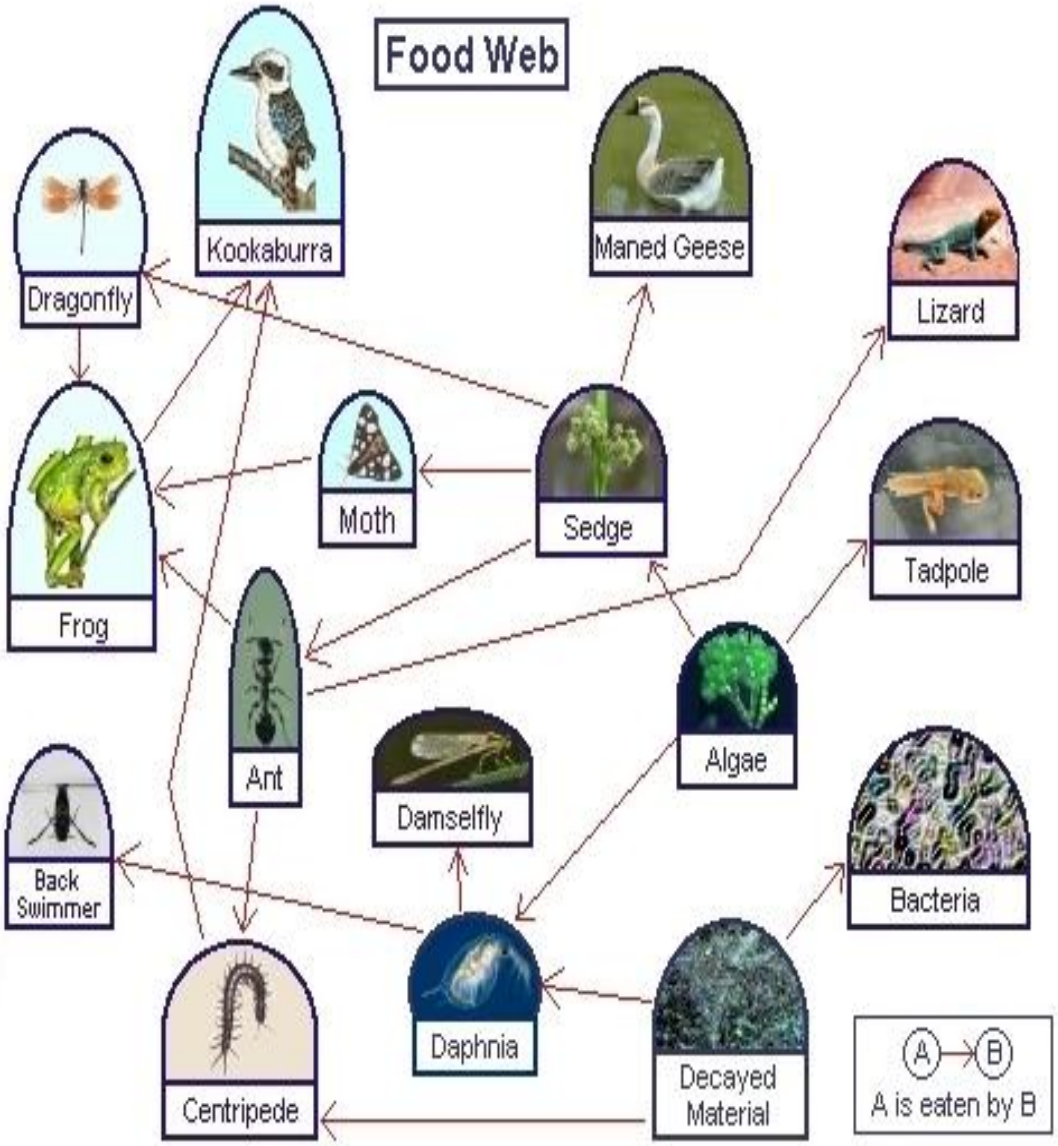


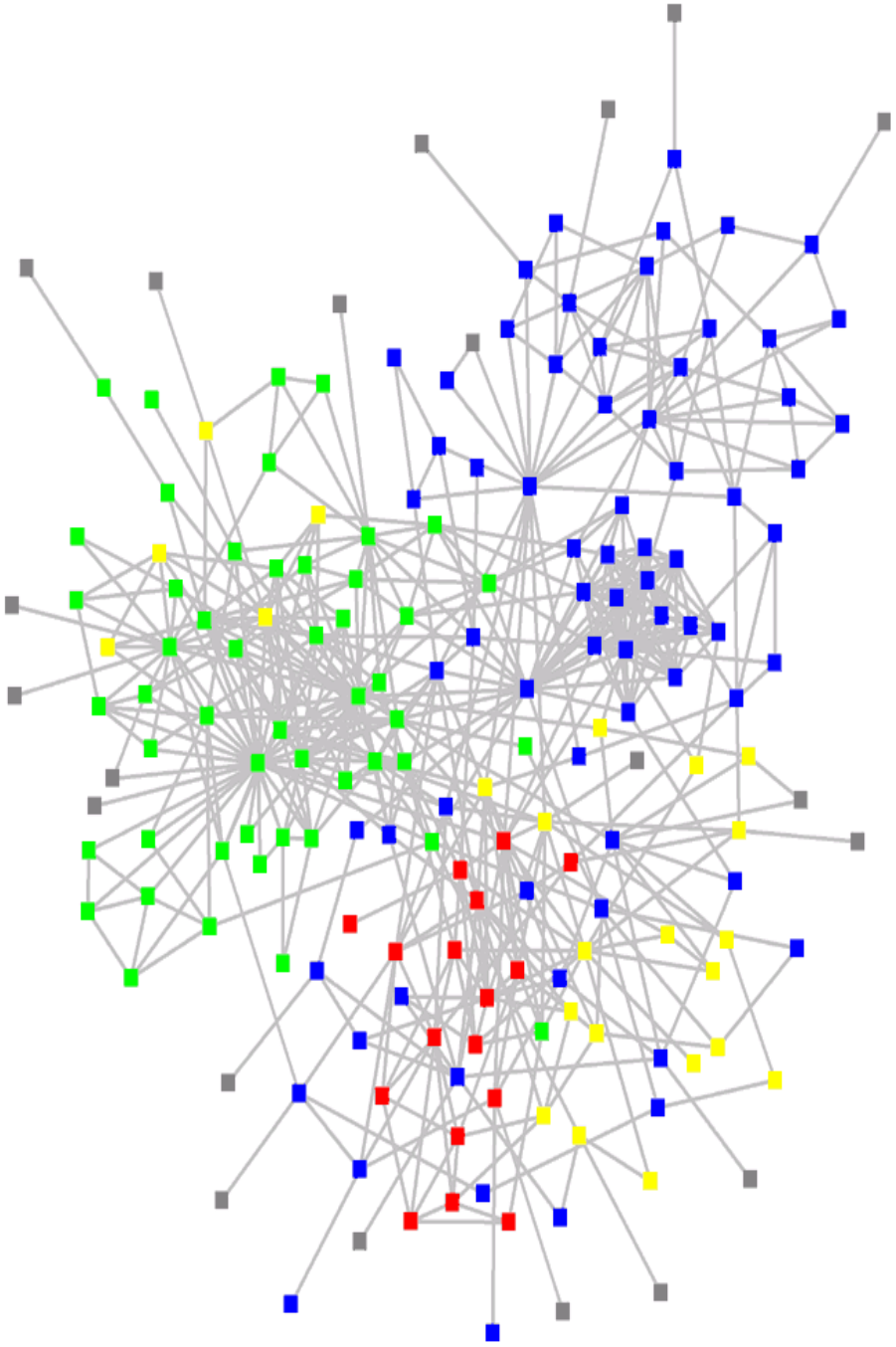
Internet Routing Paths

5 million edges

Graph Colors:
Asia Pacific - Red
Europe/Middle East/Central Asia/Africa - Green
North America - Blue
Latin American and Caribbean - Yellow
RFC1918 IP Addresses - Cyan
Unknown - White

http://www.opte.org/maps/


Food Web
Kookaburra
Maned Geese
Dragonfly
Lizard
Moth
Sedge
Frog
Tadpole
Ant
Algae
Damselfly
Back Swimmer
Bacteria
Daphnia
Centripede
Decayed Material
A
B
A is eaten by B

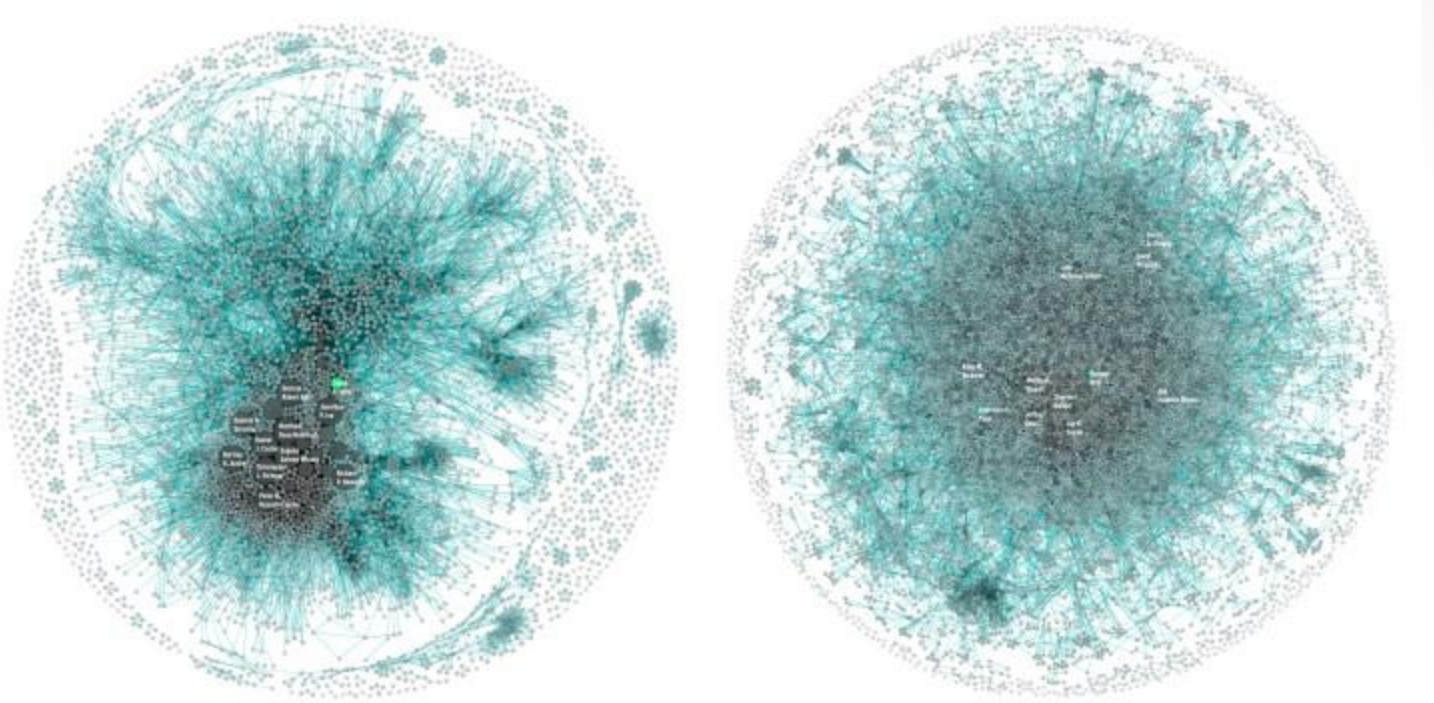


Email communication

Valdis Krebs

http://www.orgnet.com/email.html

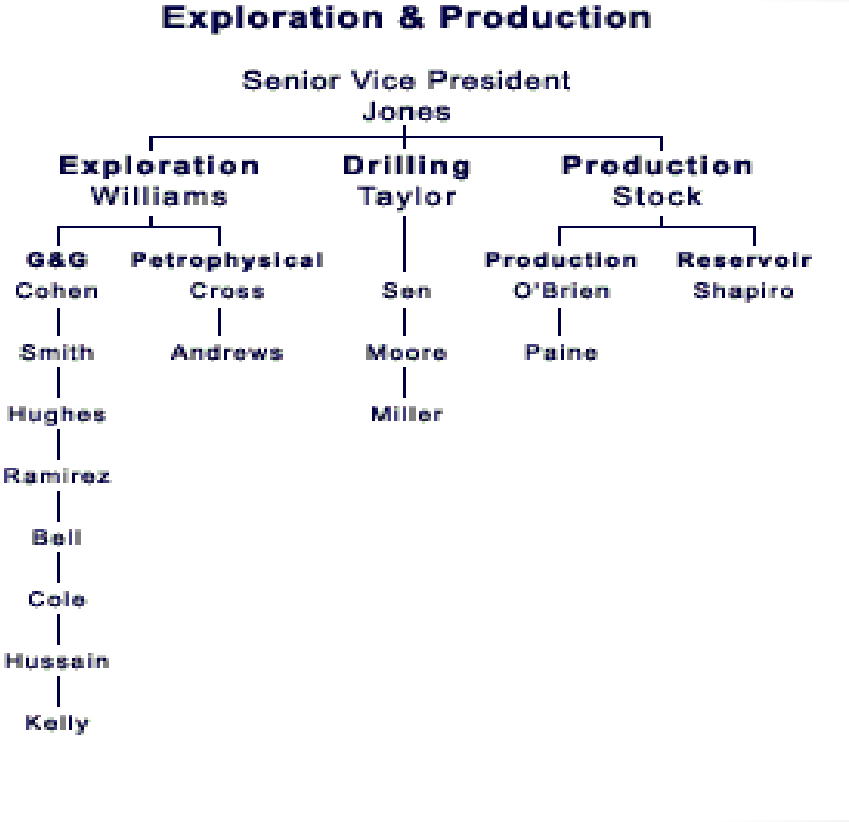
Innovation: Apple vs Google



The most notable difference we see is the presence of the group of highly connected, experienced 'super inventors' at the core of Apple compared to the more evenly dispersed innovation structure in Google

*Case Study - Network in Organization

- Consider the given Organogram of an organization
- What information can we extract from this?

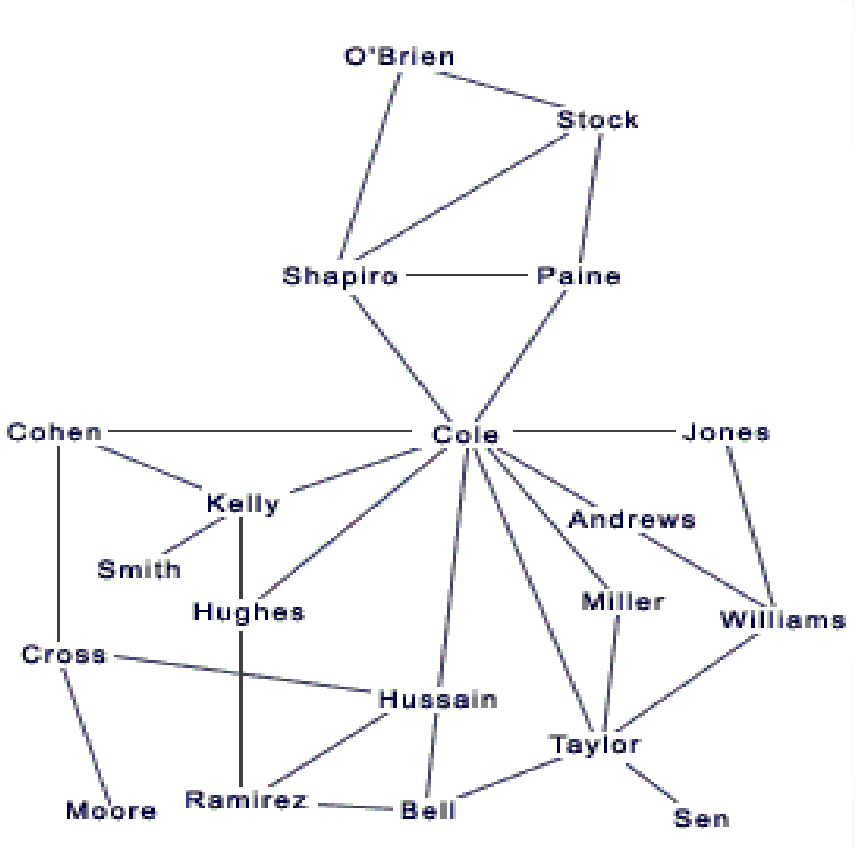


Source: http://www.robcross.org/network_ona.htm

Case Study - Network in Organization

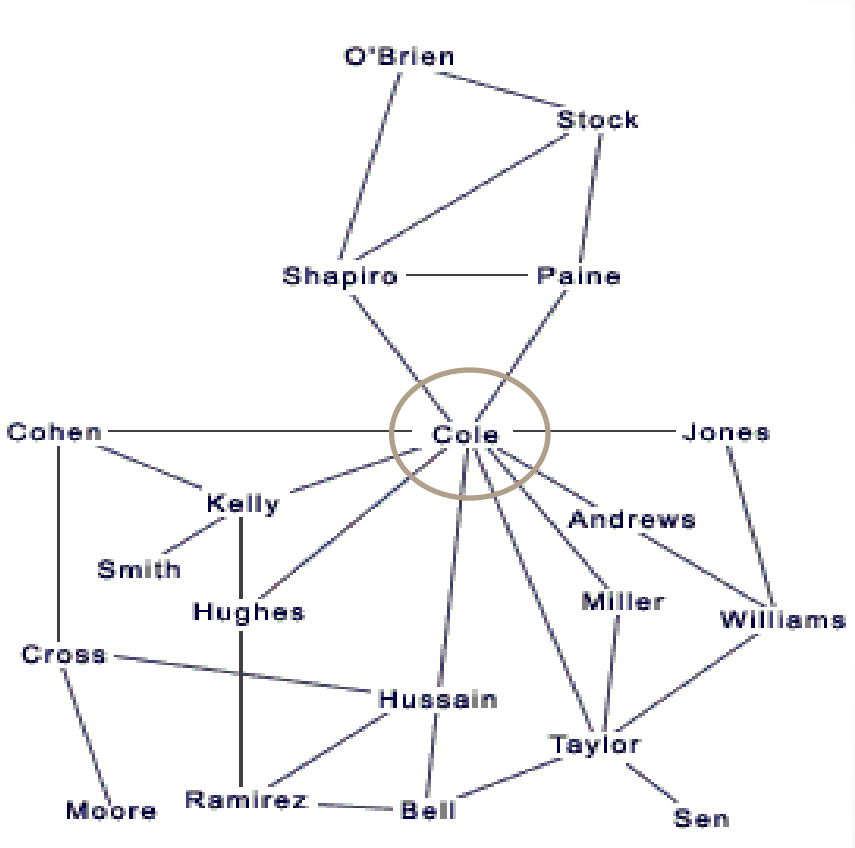
- Consider this informal network

- How did we come up with this network?



Case Study - Network in Organization

- Consider this informal network
- How did we come up with this network?
- Each edge represents communication link between two people
- What information can we extract from this network?



Network Analysis

- Network analysis can help us to find out:
  - Who are most influential / powerful people
  - Who are community leaders
  - Who are actually playing role of bridge in the network
  - Who are the most/least active people
  - Which group is more influential
  - Who is can be influenced easily
  - And much more…

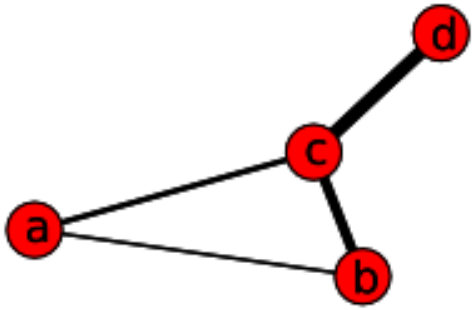
Network Analysis in Python?

NetworkX

- "Python package for the creation, manipulation and study of the structure, dynamics and functions of complex networks.”
- Data structures for representing many types of networks, or graphs
- Flexibility ideal for representing networks found in many different fields
- Easy to install on multiple platforms
- All based on Python

Quick Example

```
import networkx as nx
g = nx.Graph()
g.add_edge('a','b',weight=0.1)
g.add_edge('b','c',weight=1.5)
g.add_edge('a','c',weight=1.0)
g.add_edge('c','d',weight=2.2)
print(nx.shortest_path(g,'b','d'))
```



Graph Types

- Graph : Undirected simple (allows self loops)
- DiGraph : Directed simple (allows self loops)
- MultiGraph : Undirected with parallel edges
- MultiDiGraph : Directed with parallel edges
- can convert to undirected: g.to undirected()
- can convert to directed: g.to directed()

Adding Nodes

```
g = nx . Graph ()
g . add_node ( 'a ')
g . add_nodes_from ( [ ' b ' ,'c ' ,'d ' ])
g . add_nodes_from ( ' xyz ')
h = nx . path_graph (5)
g . add_nodes_from ( h )
g . nodes ()
```

Output:

```
[0 ,1 , ' c ' ,'b ' ,4 , ' d ' ,2 ,3 ,5 , ' x ' ,'y ' ,'z ']
```

Adding/removing Edges

- # Single edge
- g.add_edge(1,2)
- e=(2,3)

- # List of edges
- g.add_edges_from([(1 ,2) ,(1 ,3)])

- # you can remove any edge of the graph
- g.remove_edge(1,2)

Access nodes/edges

```
#all nodes
print(g.nodes())
#all edges
print(g.edges())
#number of nodes
print(g.number_of_nodes())
#number of edges
print(g.number_of_edges())
```

Nodes/edges iteration

```
#nodes iteration
for x in g.nodes():
    print(x)

#edges iteration
for y in g.edges():
    print(y)
```

Other properties

#neighbors

g.neighbors(1)

#degree

g.degree(1)

#successors

g.successors(1)

#predecessors

g.predecessors(1)

Graph Operators

- **subgraph(G, nbunch)**
  - induce subgraph of G on nodes in nbunch
- **union(G1,G2)**
  - graph union
- **disjoint_union(G1,G2)**
  - graph union assuming all nodes are different
- **cartesian_product(G1,G2)**
  - return Cartesian product graph
- **compose(G1,G2)**
  - combine graphs identifying nodes common to both
- **complement(G)**
  - graph complement
- **create_empty_copy(G)**
  - return an empty copy of the same graph class
- **convert_to_undirected(G)**
  - return an undirected representation of G
- **convert_to_directed(G)**
  - return a directed representation of G

Graph Drawing

- NetworkX is not basically graph drawing project
- IT provides basic drawing capabilities using ***matplotlib***

```
import pylab as plt
import networkx as nx
g =nx.erdos_renyi_graph(100,0.15)
nx.draw(g,with_labels=False)
plt.savefig("graph.png")
```

Graph Generators

```
#Complete Graph
nx . complete_graph (5)
# classic graphs
K_5=nx.complete_graph(5)
K_3_5=nx.complete_bipartite_graph(3,5)
lollipop=nx.lollipop_graph(10,20)
# random graphs
er=nx.erdos_renyi_graph(100,0.15)
ws=nx.watts_strogatz_graph(30,3,0.1)
ba=nx.barabasi_albert_graph(100,5)
```

DEMO

Python and NetworkX

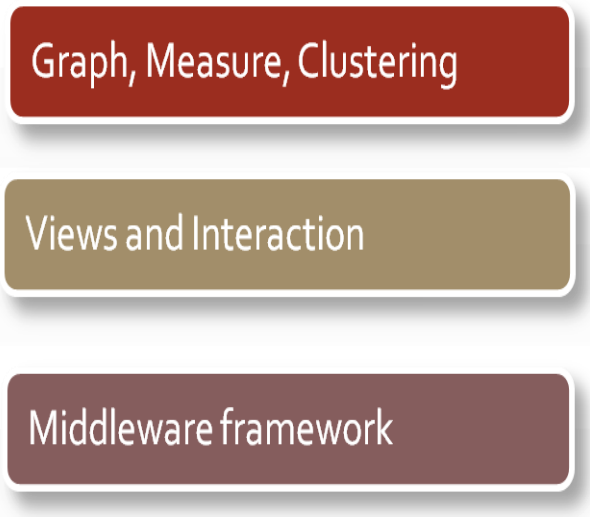
Available at:
https://github.com/mqpasta/pyconpk2018

When should I AVOID NetworkX to perform network analysis?

Large-scale problems that require faster approaches (i.e. massive networks with 100M/1B edges)

Tulip

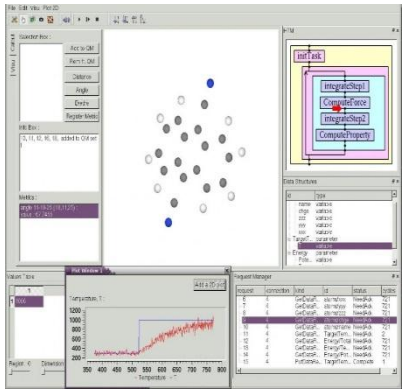
- An **information visualization** framework
- **Analysis and visualization** of relational data (such as graph)
- Provide extensive library for developers to extend by plugins
  - **Interactive** information visualization
- First released in 2001



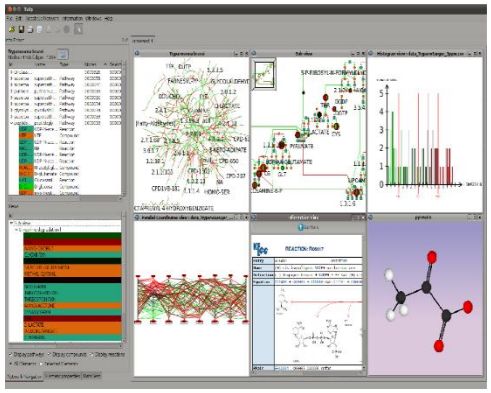
Partners

UNIVERSITY
of
GLASGOW

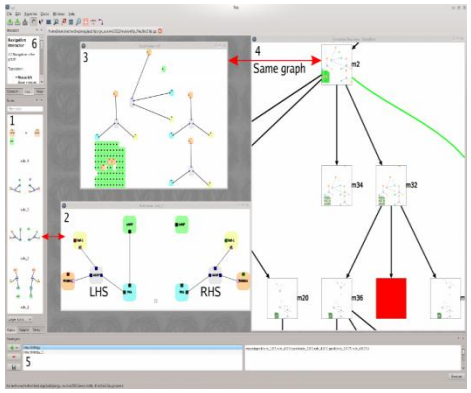
Projects using Tulip



NOSSI: New platform for parallel, hybrid quantum/classical simulations

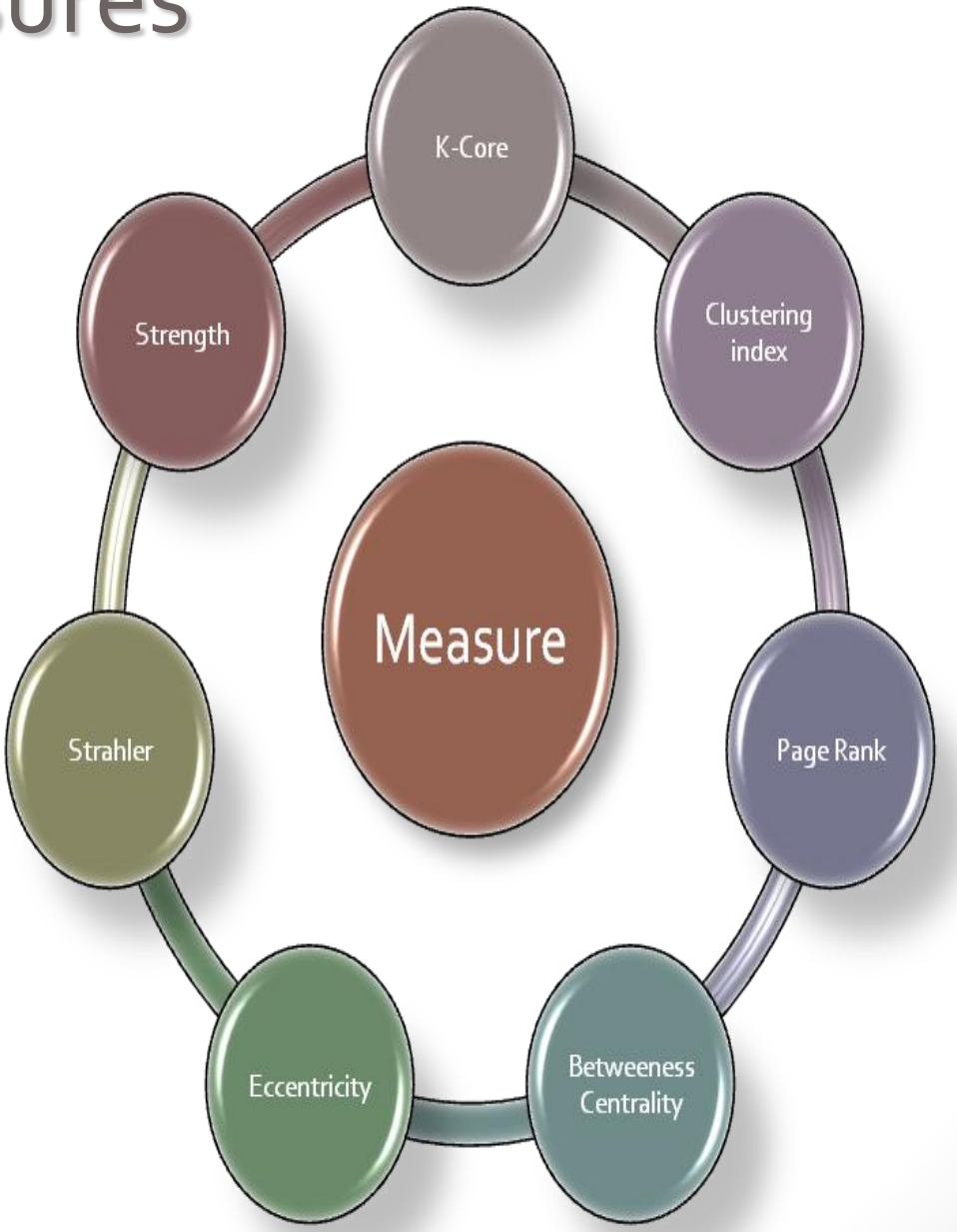


Systrip is a visual environment for the analysis of time-series data in the context of biological networks.

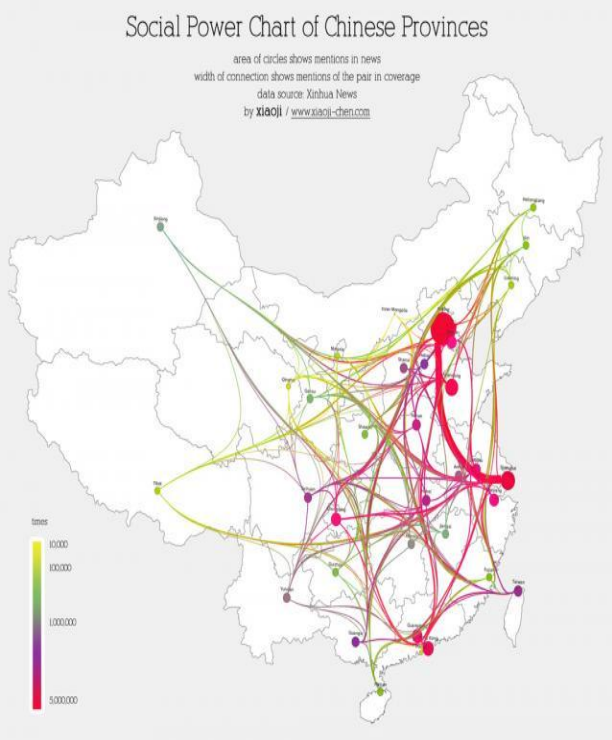


PORGY aims at designing relevant graphical representations and adequate interactions on dynamic graphs emerging from graph rewriting systems

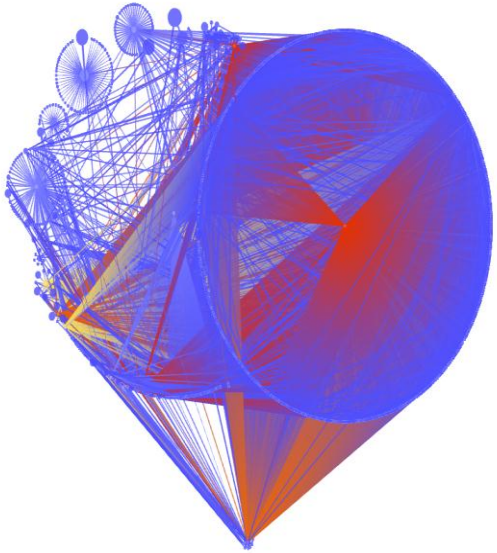
Measures

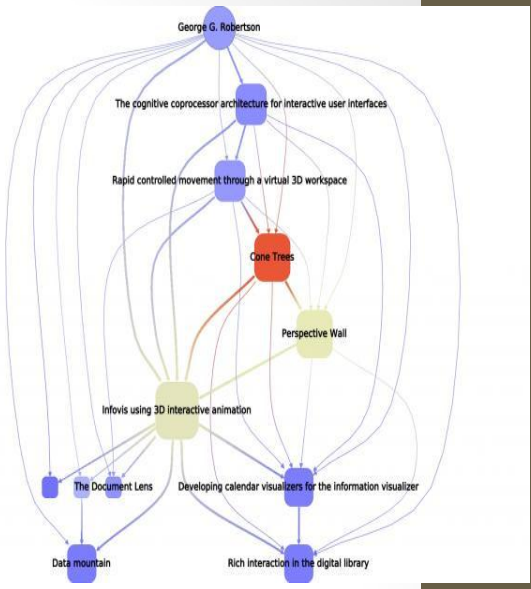

K-Core
Clustering index
Page Rank
Betweeness Centrality
Eccentricity
Strahler
Strength
Measure

Layout – Graph Drawing

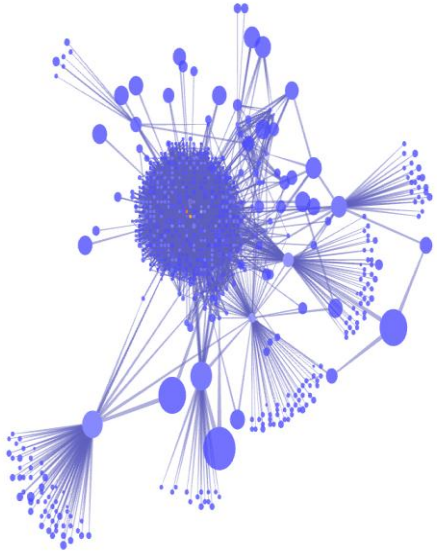


Source: http://goo.gl/DzgOsy



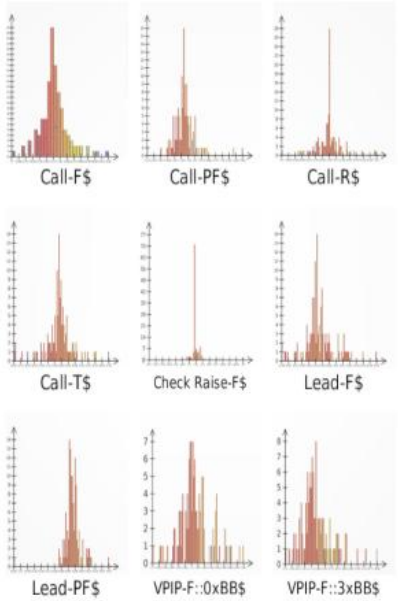


Source: http://tulip.labri.fr/TulipDrupal/?q=node/1931

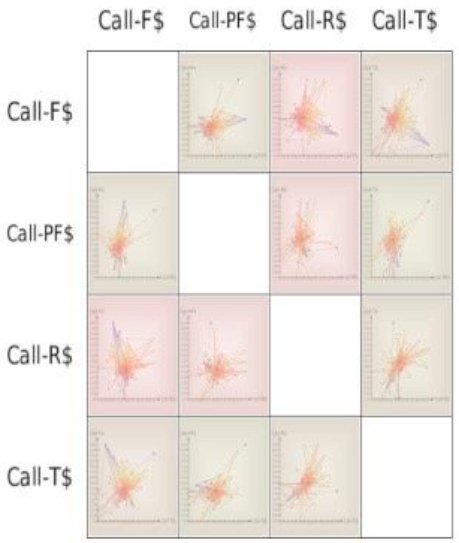


Node Link Diagram

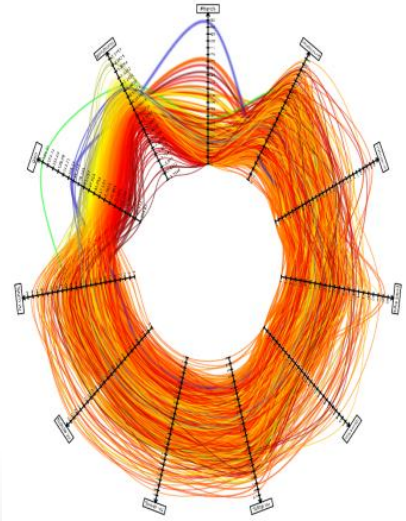
Histogram



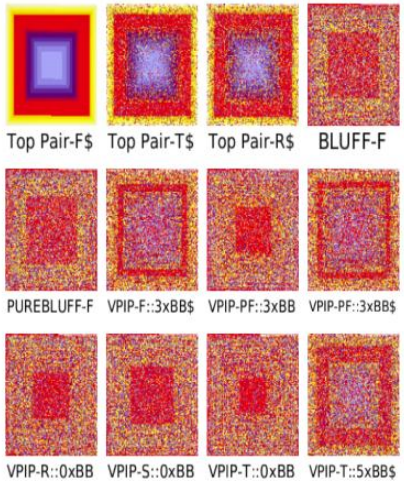
Scatter Plot



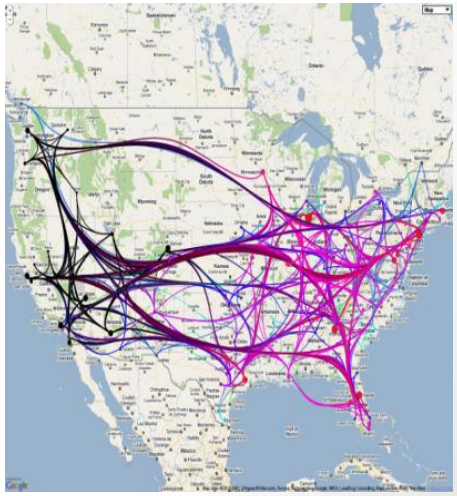
Parallel Coordinates



Pixel Oriented View

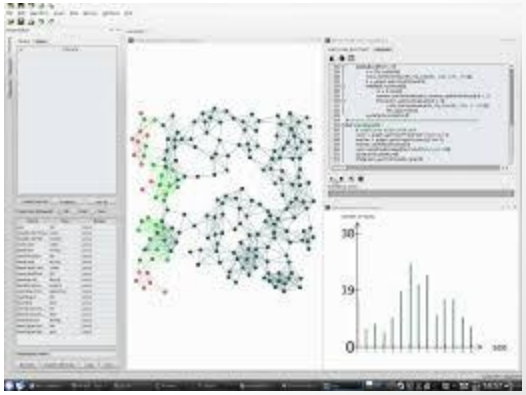


Google Map View



Tulip / Python Binding

- Tulip Python is a set of modules that exposes to Python almost all the content of Tulip C++ API
- The main features are:
  - creation and manipulation of graphs
  - storage of data on graph elements (float, integer, boolean, color, size, coordinate, list, ...)
  - application of algorithms of different types on graphs (layout, metric, clustering, ...)
  - the ability to write Tulip plugins in pure Python



DEMO

Visualization using Tulip and Python

Available at:
https://github.com/mqpasta/pyconpk2018

References

- NetworkX: Network Analysis with Python (Salvatore Scellato)
- NetworkX Tutorial Release 1.9 Aric Hagberg, Dan Schult, Pieter Swart
- NetworkX Tutorial Evan Rosen