Week 12

Agenda

Today: Network Science discussion

Networks or Graphs?

In the scientific literature the terms *network* and *graph* are used interchangeably:

Network Science	**Graph Teory**
Network	Graph
Node	Vertex
Link	Edge

Yet, there is a subtle distinction between the two terminologies: the {*network*, *node*, *link*} combination often refers to real systems: The WWW is a network of web documents linked by URLs; society is a network of individuals linked by family, friendship or professional ties; the metabolic network is the sum of all chemical reactions that take place in a cell. In contrast, we use the terms {*graph*, *vertex*, *edge*} when we discuss the mathematical representation of these networks: We talk about the web graph, the social graph (a term made popular by Facebook), or the metabolic graph. Yet, this distinction is rarely made, so these two terminologies are often synonyms of each other.

History

In the previous era, it was difficult to gather or uncover data.

1950’s Random Graphs (Paul Erdos)

Choices independent of current network

1960’s Small world experiment (Stanley Milgram)

- https://en.wikipedia.org/wiki/Small-world_experiment.
- Average path length is 5.5-6.
- FB has avg path length of 4.75, twitter 4.67, MS Messenger 6.6

-> Small world (Watts-Strogatz) graphs: creates rings with clusters

Book recommendation: Stanley Wasserman and Katherine Faust: "Social network Analysis"

- http://www.amazon.com/Social-Network-Analysis-Applications-Structural/dp/0521387078/x

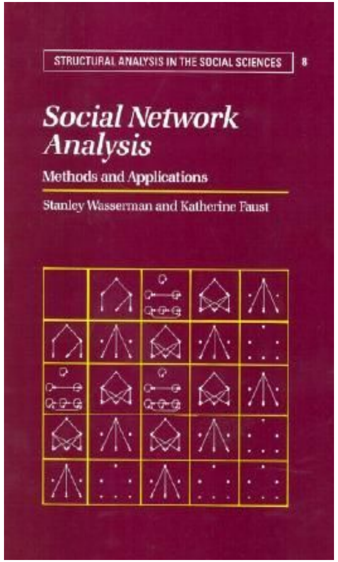
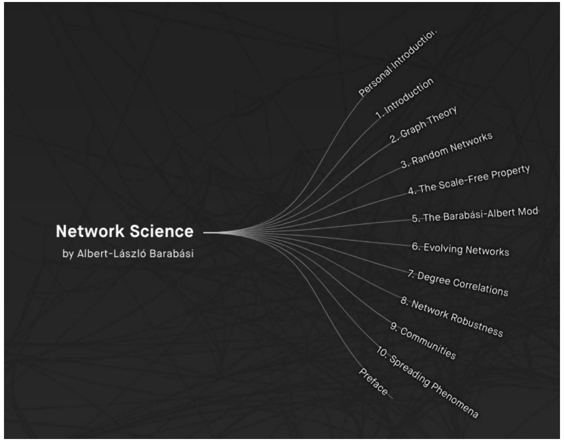


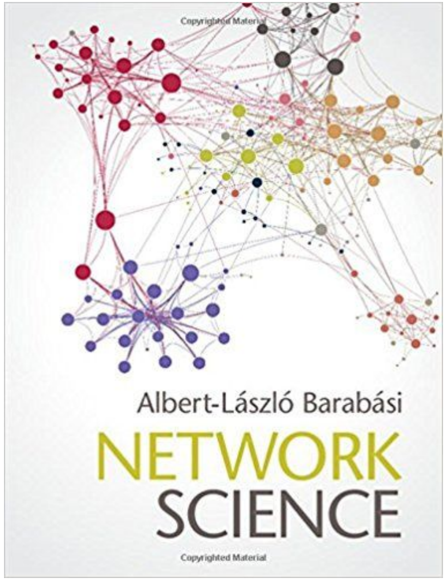
Table of Contents

Recent History

2000's Scale-free (follows power law) networks

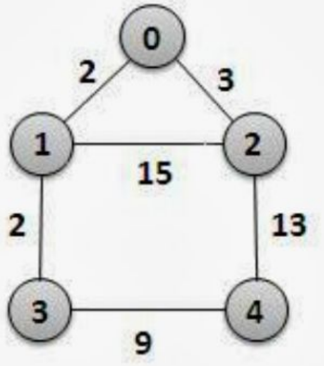
- Long tail. Fault tolerant (random attacks). Hubs.
- Examples: www, travel/airlines (and disease), protein interactions, social networks
- Preferential attachment
- Scale-free (Barabasi): http://arxiv.org/pdf/cond-mat/0106096.pdf and https://barabasi.com/f/623.pdf





networksciencebook.com

Dense Representation



	0	1	2	3	4
0	0	2	3	0	0
1	2	0	15	2	0
2	3	15	0	0	13
3	0	2	0	0	9
4	0	0	13	9	0

Adjacency Matrix Representation of Weighted Graph

Dense Representation: Adjacency matrix

- Vertex-Vertex
- 0/1 or weight

Sparse Representation

- Version 1: V1:[V2,V3]
- Version 2: V1:V2:nonzero-weight:node1size:node1color

How might a sparse graph be represented?

Statistics & Algorithms

General Language

- Graph theory vs Network Science
- Vertices/nodes (N), edges (E)
- Path (L), hops, cycles
- Undirected, directed, DAG (directed acyclic graph), Bipartite
- Connectivity, component
- Hypergraphs

Node Level

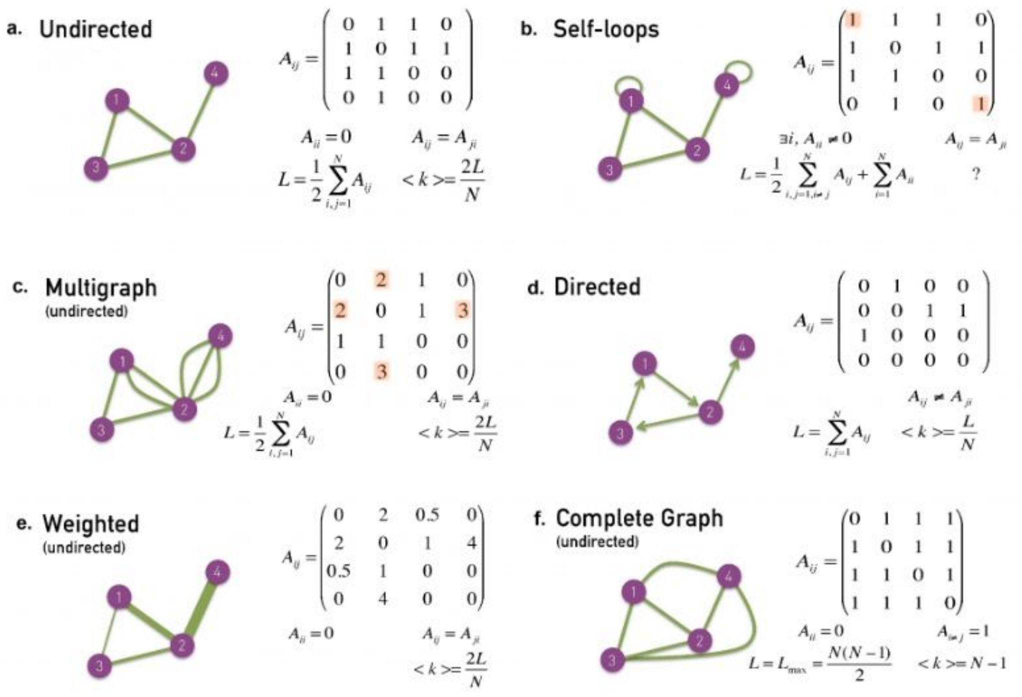
- Size: (N,E) (sum rows, columns in adj matrix)
- Degree: (k), and in-degree or out-degree

Edge Level

- Weight

Graph Level

- Avg degree: 2 * E/N (<k>)
- Avg path length
- Diameter (longest shortest path between any two nodes)
- Density (ratio of edges to possible edges, 2e / (N*(N-1)))



Statistics & Algos

General Language

- Graph theory vs Network Science (not hard and fast)
- Vertices/nodes (N), edges (E)
- Path (L), hops, cycles
- Undirected, directed, DAG, Bipartite
- Connectivity, cuts, component
- Hypergraphs

Node Level

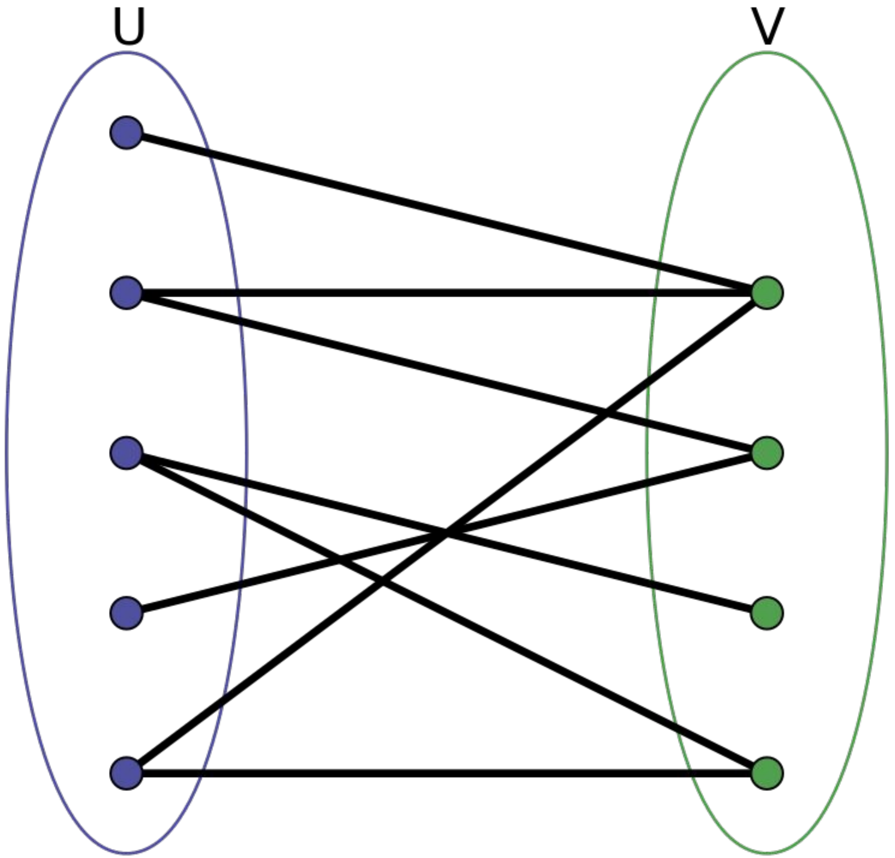
- Size: (N,E) (sum rows, columns in adj matrix)
- Degree: (k), and in-degree or out-degree

Edge Level

- Weight

Graph Level

- Avg degree: 2 * E/N (<k>)
- Avg path length
- Diameter (longest shortest path)
- Density (ratio of edges to possible edges, 2e / (N*(N-1)))



Statistics & Algos: Clustering

Clustering measures
Partition a graph into natural groups so that the nodes in the same cluster are more close to each other than to those in other clusters.

Many algorithms for clustering exist: Min-Cut Tree, K-means, Greedy Merge, random walks, etc.

How to know if a cluster is "good"?

- Clustering Coefficient, number of intra-/inter-edges, etc
- e.g. for a node (2e / (k*(k-1))) where k is number of neighbors, and e is number of connections between neighbors
- e.g. 3 * number of triangles / number of connected triplets (Wasserman)

Connected components
find the "hard" clusters, sub graphs that are not connected to the rest of the graph

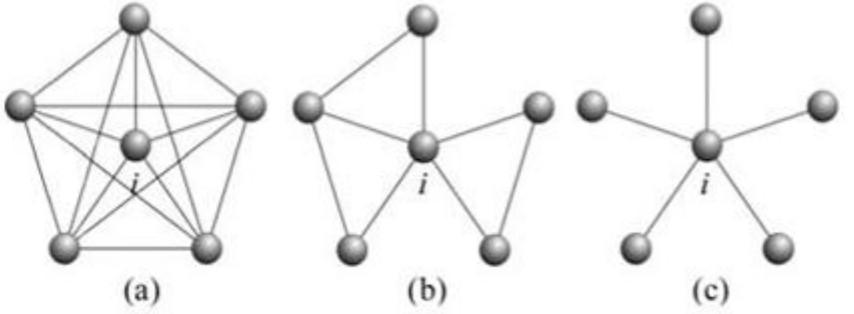


Figure 4 - Example of three networks and respective clustering coefficients (see Eq. (1)). In (a), $cc_i = \frac{10(2)}{5(4)} = 1$ (the vertices around i are fully connected), (b) $cc_i = \frac{3(2)}{5(4)} = 0.3$ and (c) $cc_i = \frac{0(2)}{5(4)} = 0$. The maximum number of edges among the neighbors of i is given by $k_i(k_i - 1)/2$.

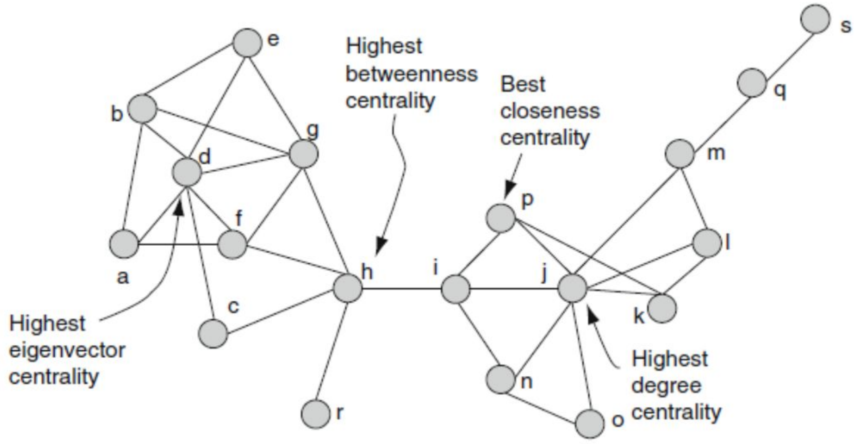
Statistics & Algos - Centrality: how important is a node

Centrality measures

- https://ocw.mit.edu/courses/civil-and-environmental-engineering/1-022-introduction-to-network-models-fall-2018/lecture-notes/MIT1_022F18_lec4.pdf
- Degree Centrality
- Closeness Centrality (1 / total distance)
- Betweenness Centrality (how many shortest paths pass through a node)
- Pagerank / Eigenvector

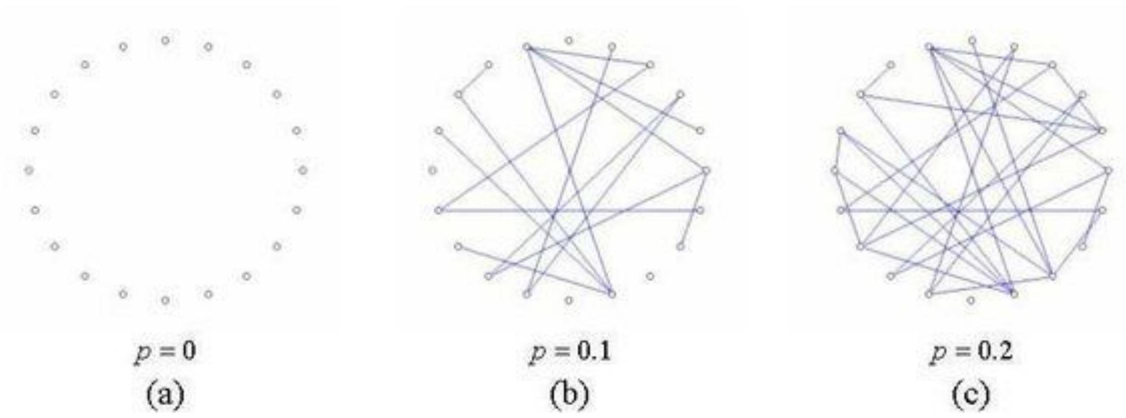
Pagerank

- https://graui.de/pageRank.htm
- https://medium.com/@sarthakanand/page-rank-b7072c61dd85
- PR(A) = (1-d) + d (PR(T1)/C(T1) + … + d(PR(Tn)/C(Tn)))
- C(X) is the vote
- PR(x) is the importance of the vote
- d is dampening
- Why hard to do in (original) MapReduce?



Breakout!

Growth Modeling - Erdos Random Graph



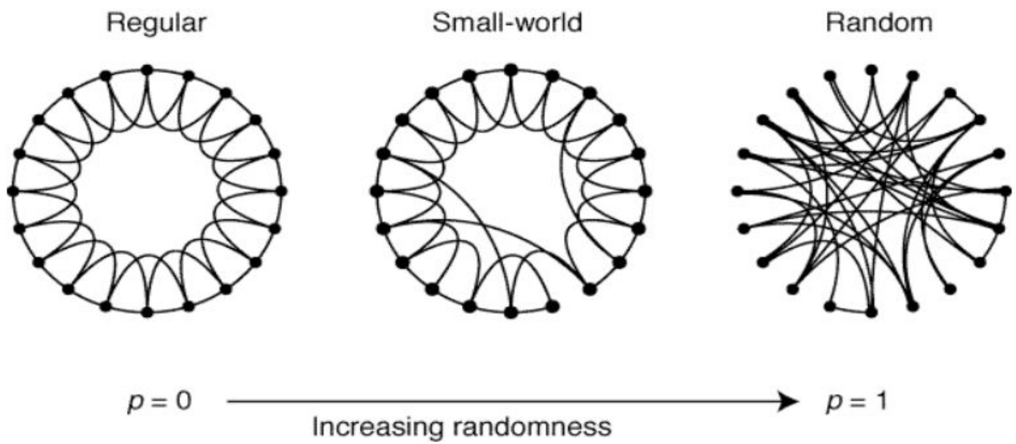
Erdos

- Review algorithm
- Connect a set of nodes with uniform probability (p)

Descriptive Statistics

- Properties depend on (Np) around value <k>
- Degree distribution is Poisson, giant component for large (N)
- Does not create hubs, triangle closures, Clustering coefficient approaches 0

Growth Modeling - Watts-Strogatz



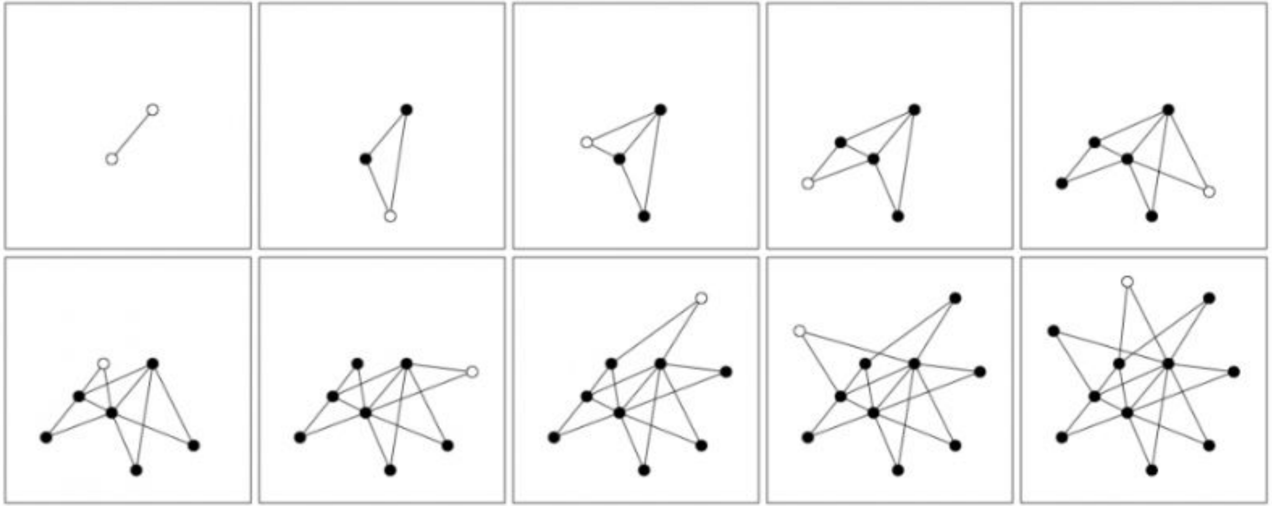
Watts-Strogatz

- Algorithm
- Every node has same number of edges in a ring lattice
- Random rewiring with fixed probability

Descriptive Statistics

- Creates locally clustered graph

Growth Modeling - Barabasi

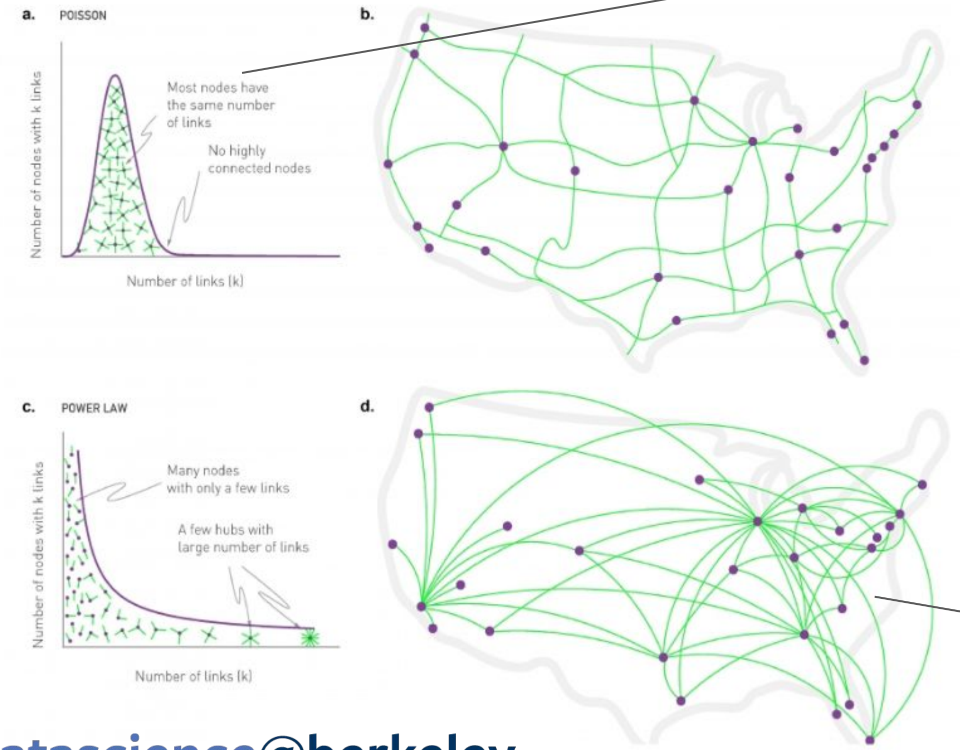


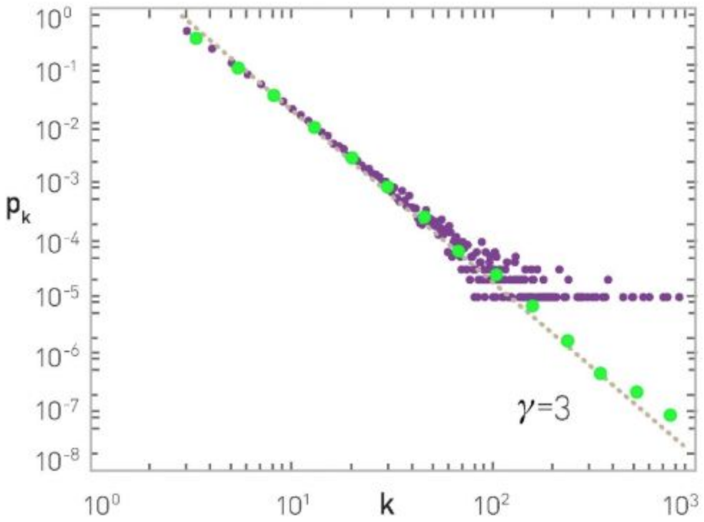
Barabasi Preferential Attachment

- p(k) is probability that a node has degree k
- log scale / power law (p(k) ~ k^-3)

Growth Modeling - Barabasi

Random graph





Barabasi

Graph libraries

https://spark.apache.org/docs/2.1.0/graphx-programming-guide.html

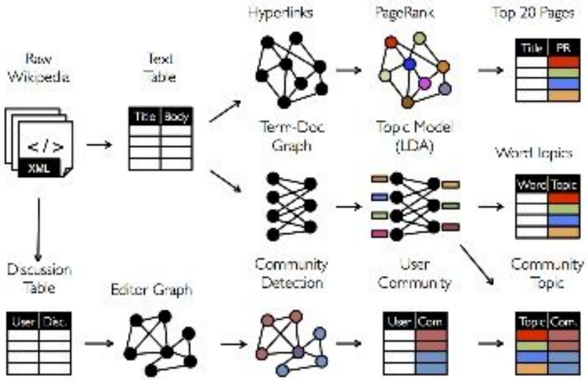
GraphX

Scala.io

GraphX (Apache Spark)

Offers a Graph API on top of Spark.

Enabling cross-world manipulations



ebiznext

https://networkx.github.io/

NetworkX (Python)

Contact

Mailing list
Issue tracker
Source

Releases

Stable (notes)

2.5 — August 2020
download | doc | pdf

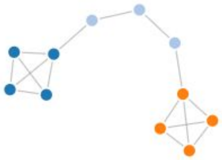
Latest (notes)

2.6 development
github | doc | pdf

Archive

NetworkX
Network Analysis in Python

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



Software for complex networks

- Data structures for graphs, digraphs, and multigraphs
- Many standard graph algorithms
- Network structure and analysis measures
- Generators for classic graphs, random graphs, and synthetic networks
- Nodes can be "anything" (e.g., text, images, XML records)

https://graph-tool.skewed.de/

Graph-tool (Python written in C++)

graph-tool | Efficient network analysis

Download | Documentation | Mailing List | Git | Issues

What is graph-tool?

Graph-tool is an efficient Python module for manipulation and statistical analysis of graphs (a.k.a. networks). Contrary to most other Python modules with similar functionality, the core data structures and algorithms are implemented in C++, making extensive use of template metaprogramming, based heavily on the Boost Graph Library. This confers it a level of performance that is comparable (both in memory usage and computation time) to that of a pure C/C++ library.

Download version 2.35

Installation instructions | Changelog

Conda installation (GNU/Linux | MacOS)

```
conda create --name gt -c conda-forge graph-tool
conda activate gt
```

It is *Fast!*

Despite its nice, soft outer appearance of a regular Python module, the core algorithms and data structures of graph-tool are written in C++, with

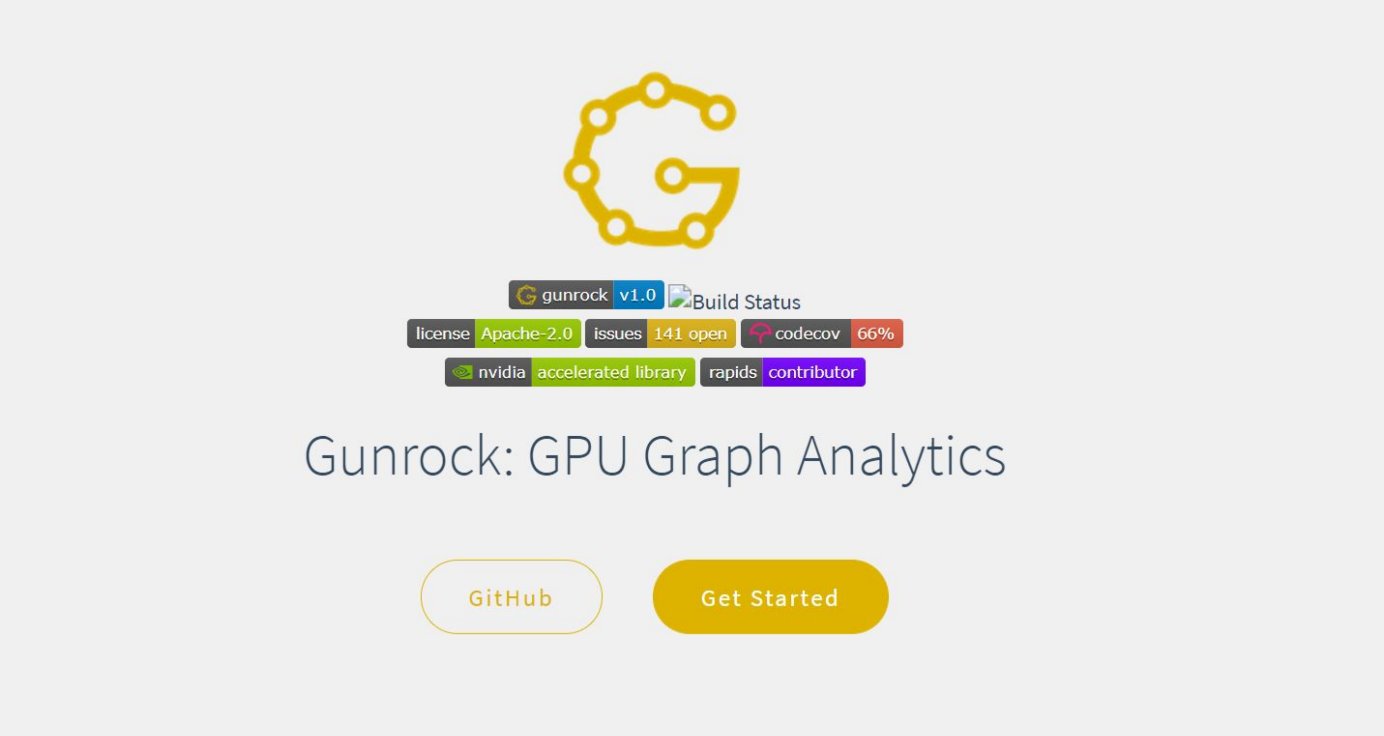
Extensive Features

An extensive array of features is included, such as support for arbitrary vertex, edge or graph properties, efficient "on the fly" filtering of vertices and edges, powerful

Powerful Visualization

Conveniently draw your graphs, using a variety of algorithms and output formats (including to the screen). Graph-tool has its own layout algorithms and versatile, https://gunrock.github.io/docs/#/gunrock/overview

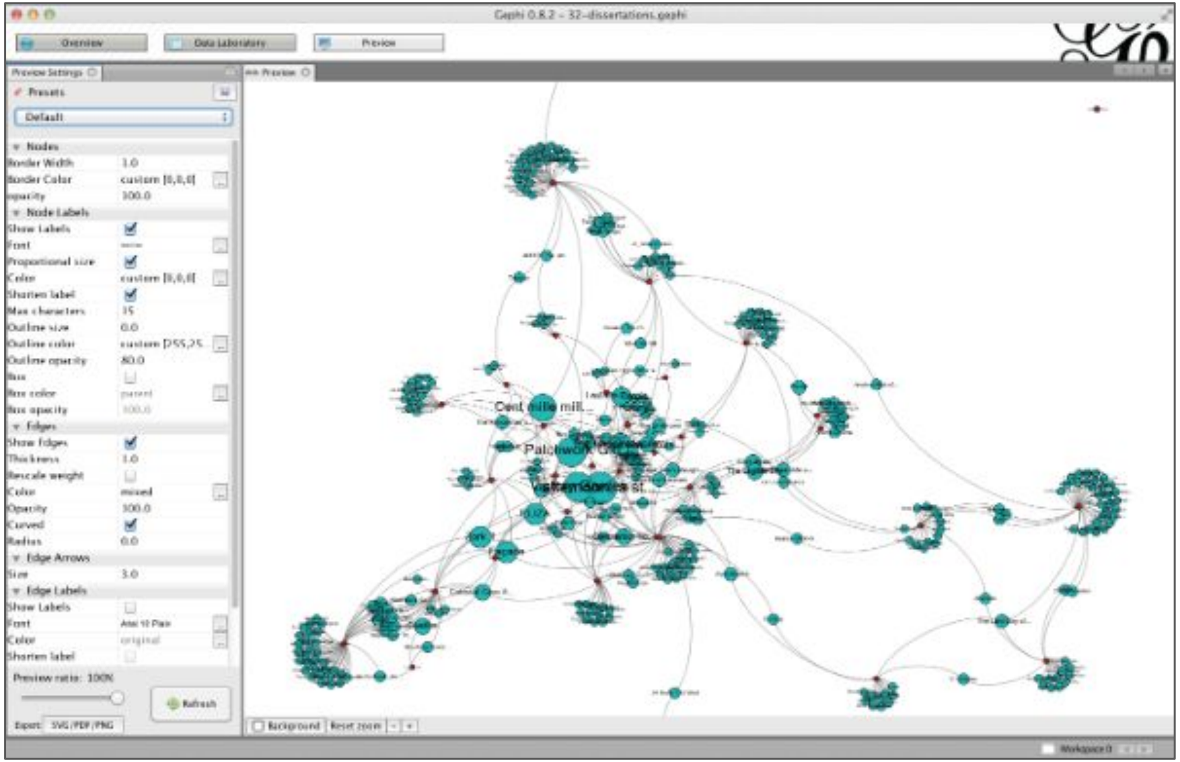
Gunrock - for GPUs



https://gephi.org/

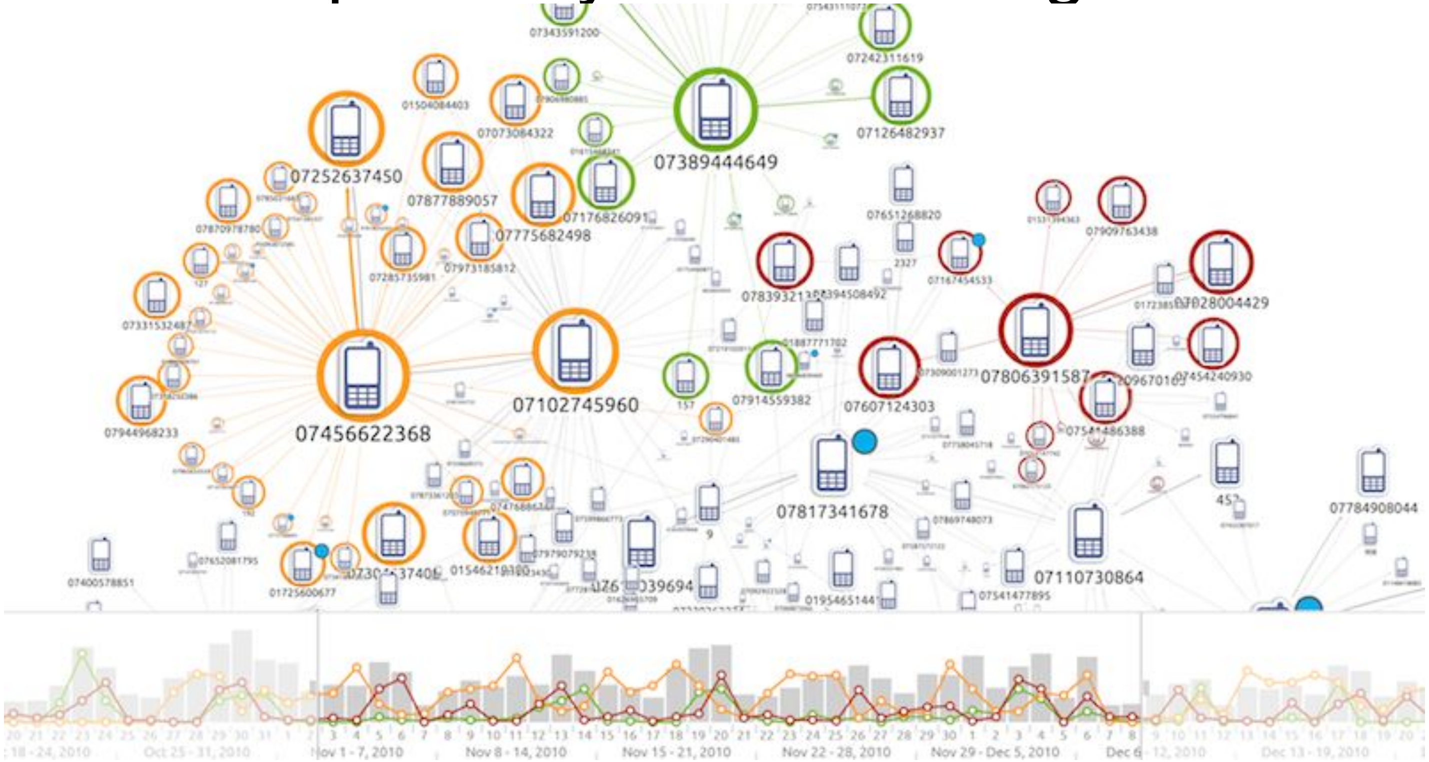
Gephi

Open Graph Viz Platform



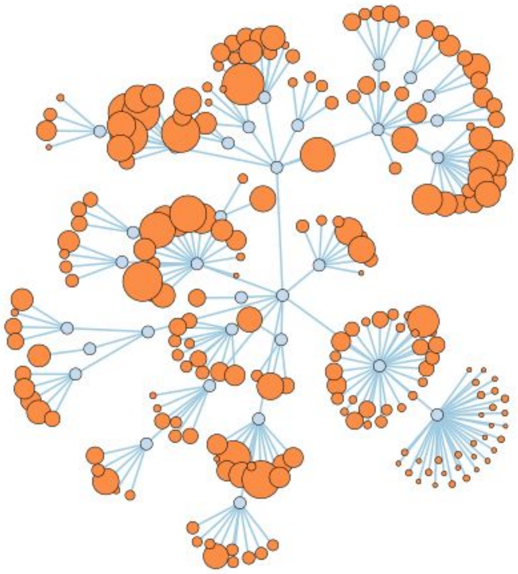
https://github.com/d3/d3

D3 - JavaScript library for visualizing data



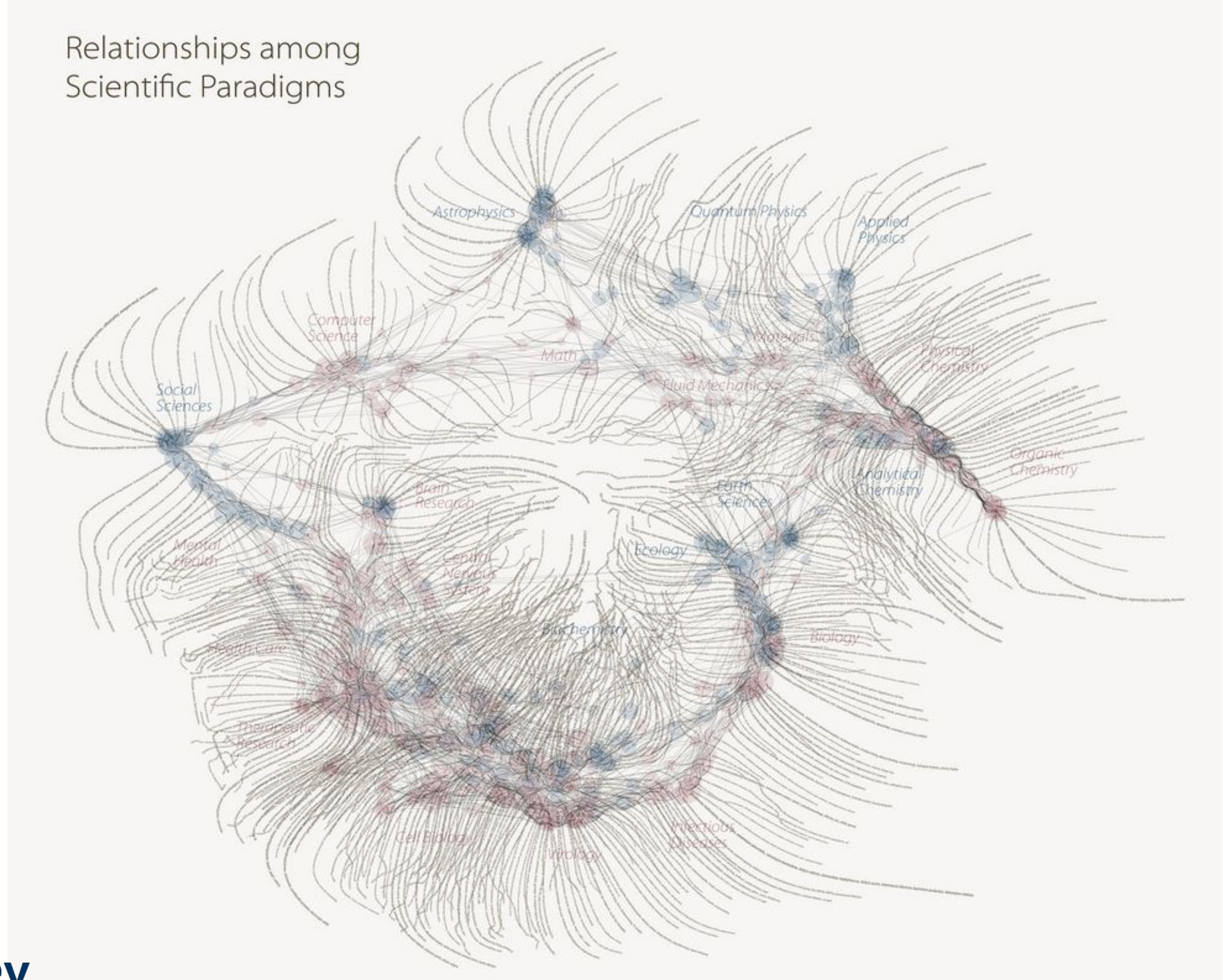
D3 - “Force Directed”

"physical tools" - e.g., include physical forces



http://mbostock.github.io/d3/talk/20111116/force-collapsible.html

Large Scale Visualization



http://wbpaley.com/brad/mapOfScience/scienceMapFullColorPrint2_lowRes_b.jpg