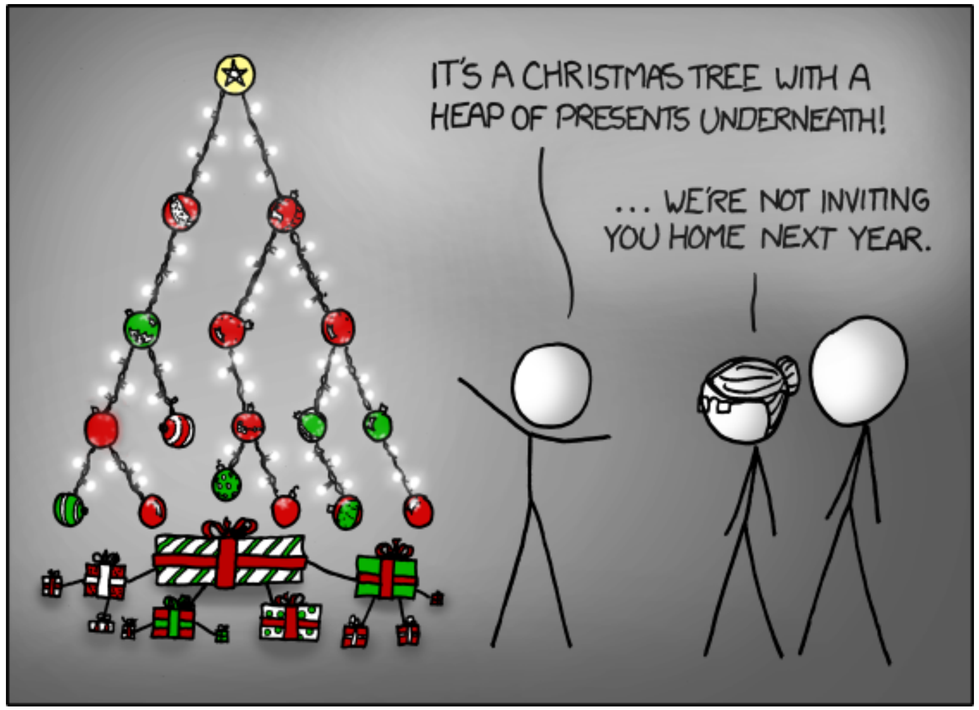
COMP-5360 / MATH-4100
Intro Data Science
Networks

Bei Wang and Anna Little

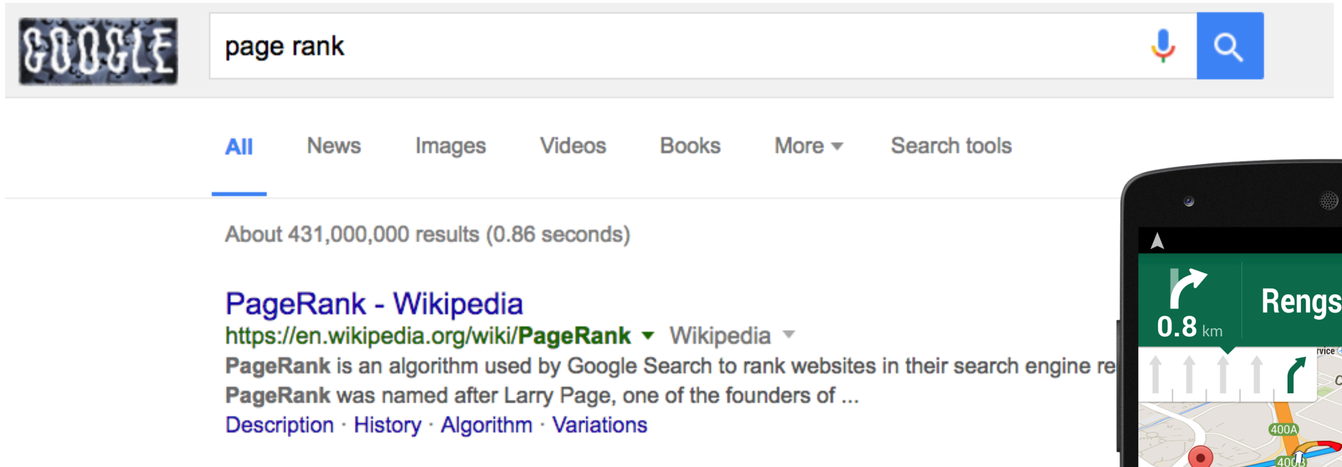
Source: Alexander Lex

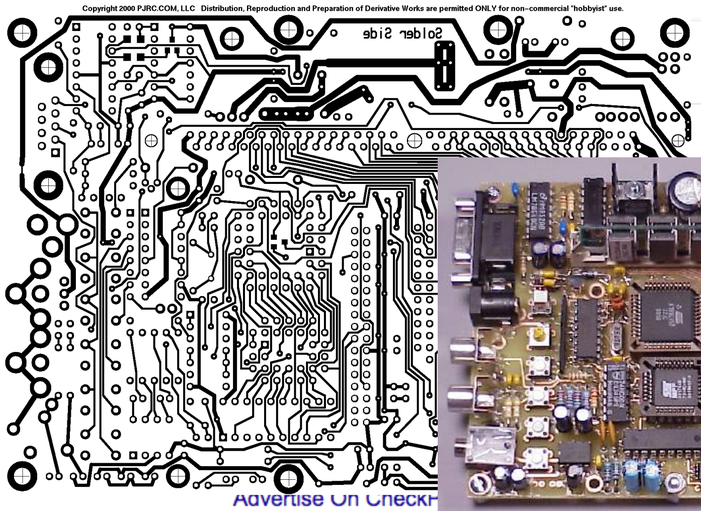


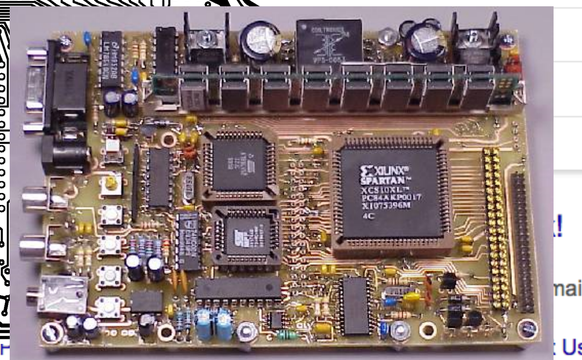
[xkcd]

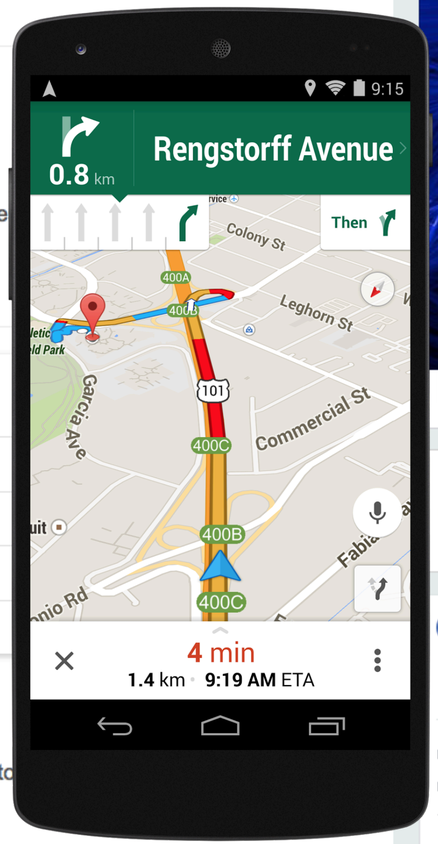
Applications of Graphs

Without graphs, there would be none of these:`










facebook
December 2010

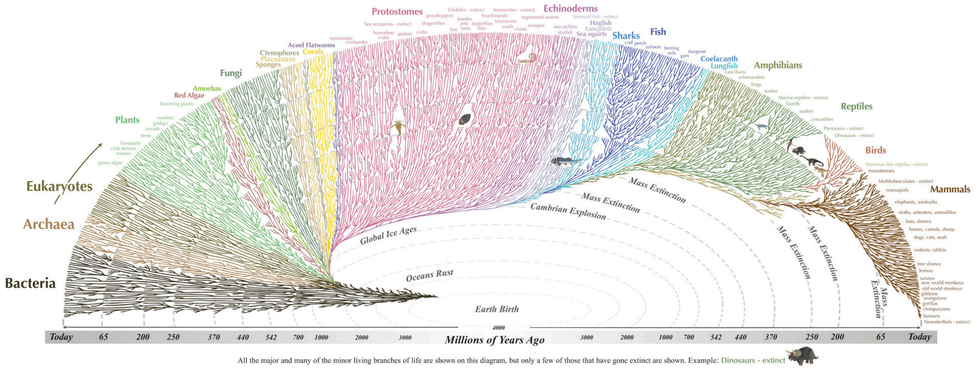
Biological Networks

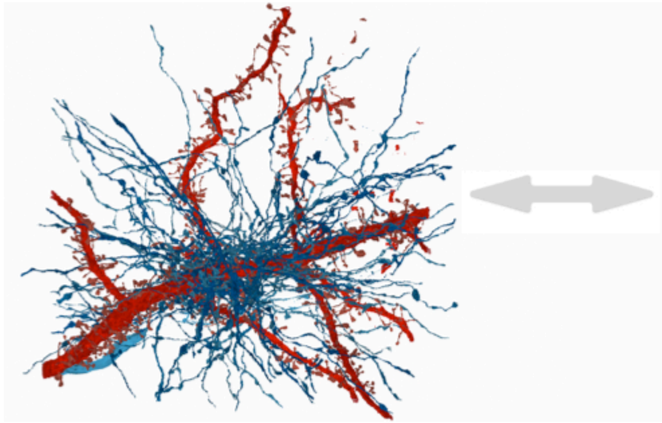
Interaction between genes, proteins and chemical products

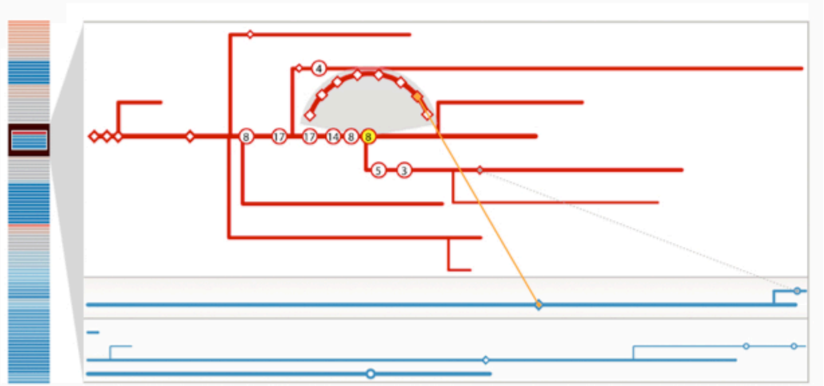
The brain: connections between neurons

Your ancestry: the relations between you and your family

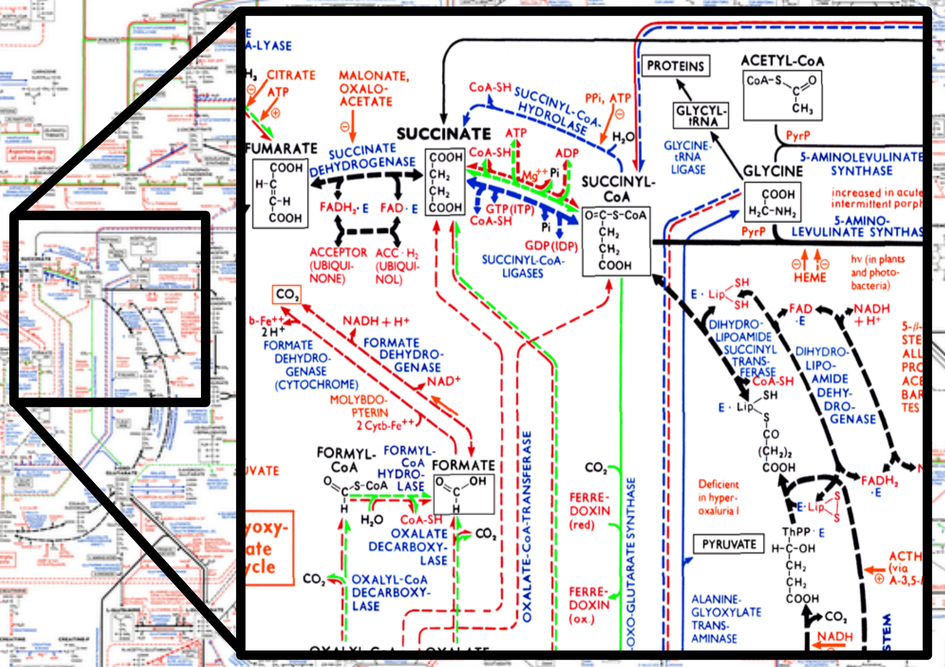
Phylogeny: the evolutionary relationships of life







[Beyer 2014]


CITRATE
ATP
MALONATE, OXALO-ACETATE
FUMARATE
SUCCINATE DEHYDROGENASE
SUCCINATE
CoA-SH
SUCCINYL-CoA HYDROLASE
PPi, ATP
H2O
ATP
ADP
PROTEINS
ACETYL-CoA
GLYCYL-tRNA
GLYCINE tRNA LIGASE
GLYCINE
PyrP
5-AMINOLEVULINATE SYNTHASE
SUCCINYL-CoA
GTP(ITP)
GDP(IDP)
SUCCINYL-CoA-LIGASES
FADH2 · E
FAD · E
ACCEPTOR (UBIQUI-NONE)
ACC · H2 (UBIQUI-NOL)
CO2
NADH + H+
FORMATE DEHYDRO-GENASE (CYTOCHROME)
FORMATE DEHYDRO-GENASE
NAD+
MOLYBDO-PTERIN
2 Cytb-Fe++
FORMYL-CoA
FORMYL-CoA HYDRO-LASE
FORMATE
H2O
OXALATE DECARBOXY-LASE
OXALYL-CoA DECARBOXY-LASE
FERRE-DOXIN (red)
FERRE-DOXIN (ox.)
OXALATE-CoA-TRANSFERASE
OXO-GLUTARATE SYNTHASE
HEME
hv (in plants and photo-bacteria)
DIHYDRO-LIPOAMIDE SUCCINYL TRANS-FERASE
DIHYDRO-LIPO-AMIDE DEHY-DRO-GENASE
Deficient in hyper-oxaluria I
PYRUVATE
ALANINE-GLYOXYLATE TRANS-AMINASE
NADH

Michal 2000

Graph Analysis Case Study

Nicholas Christakis, Yale

The hidden influence of social networks

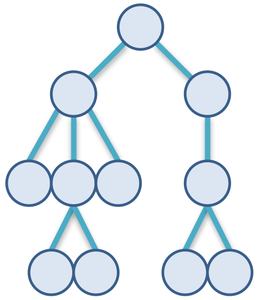
Talks about how a variety of traits, such as happiness or obesity, can spread from person to person

https://www.ted.com/talks/nicholas_christakis_the_hidden_influence_of_social_networks

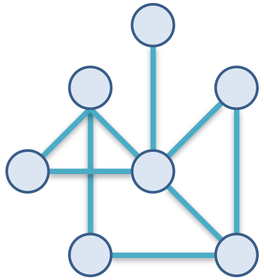
Graph Theory Fundamentals

See also “Network Science”, Barabasi
http://barabasi.com/networksciencebook/chapter/2

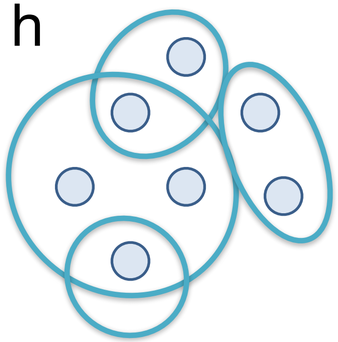
Tree



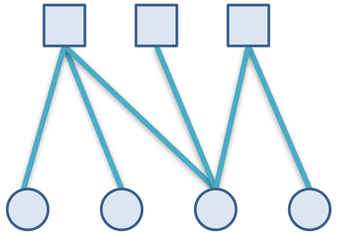
Network



Hypergraph

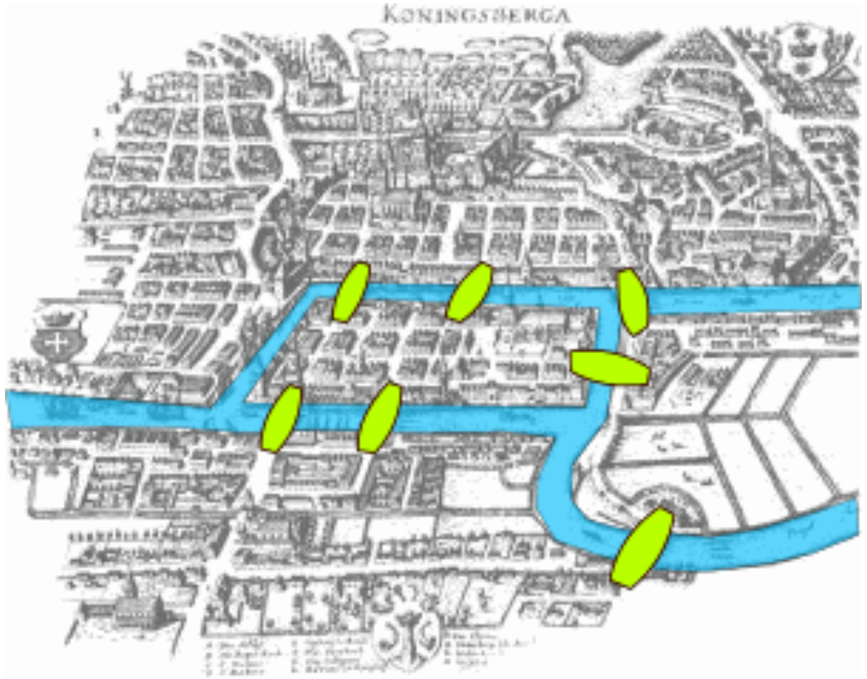


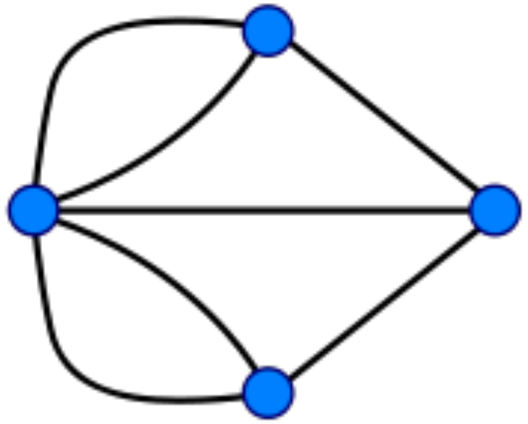
Bipartite Graph



Königsberg Bridge Problem (1736)

Can you take a walk and visit every land mass without crossing a bridge twice?





Leonhard Euler:
Only possible with a graph with at most two nodes with an odd number of links.
This graph has four nodes with odd number of links.

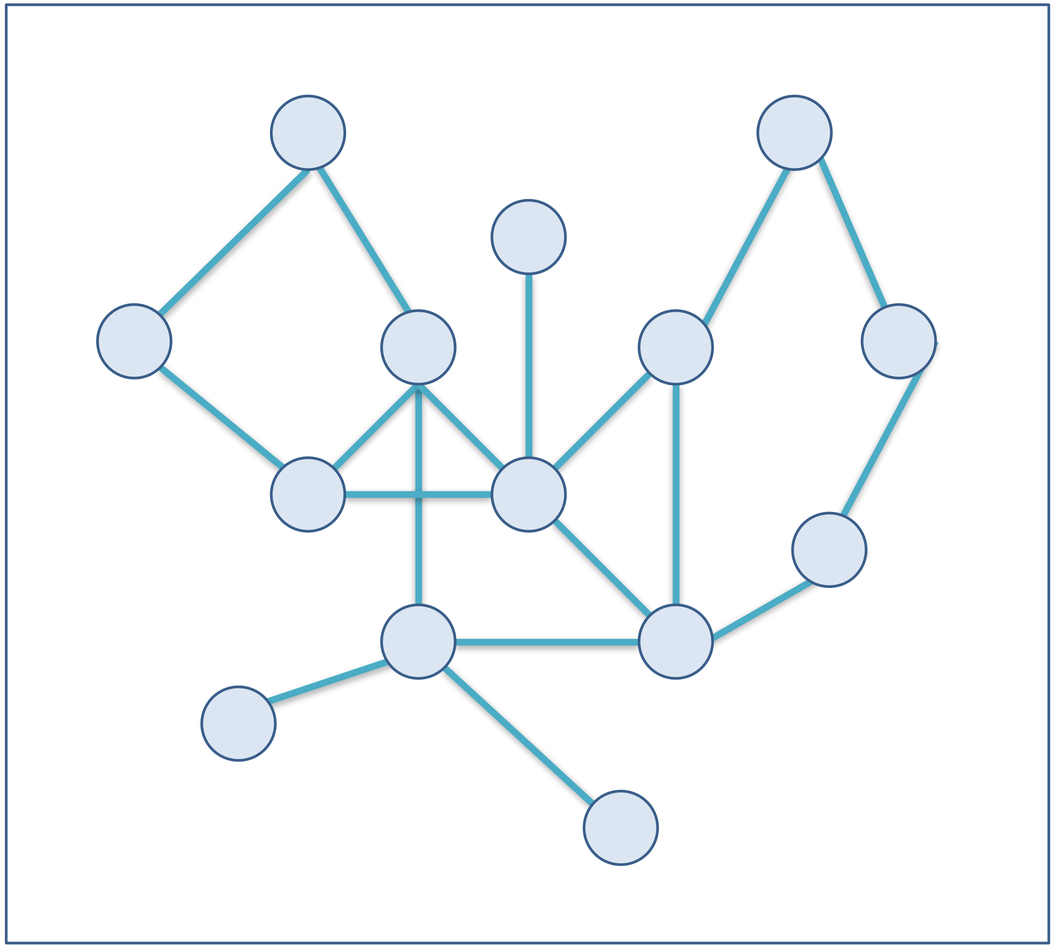
http://barabasi.com/networksciencebook/chapter/2#bridges

Graph Terms

A graph **G(V,E)** consists of a set of **vertices V** (also called nodes) and a

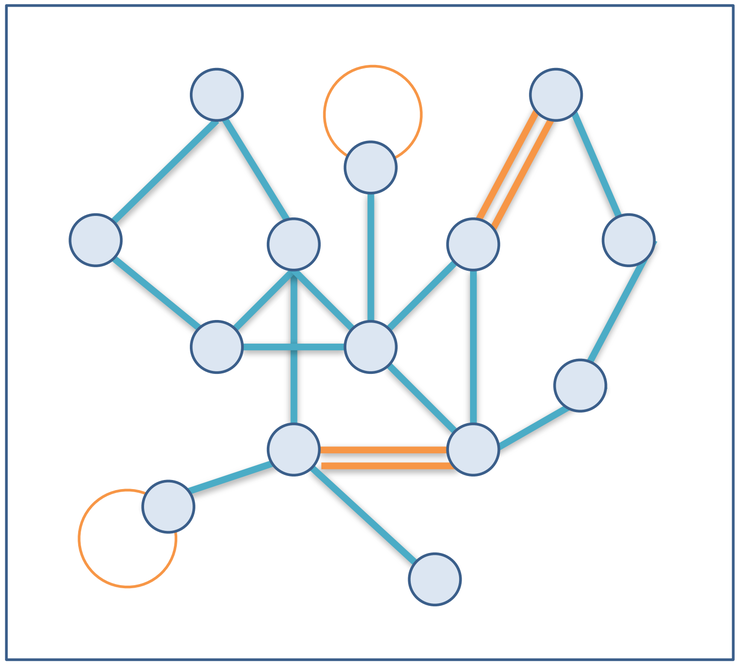
set of **edges E** (also called links) connecting these vertices.

Graph and **Network** are often used interchangeably



Graph Term: Simple Graph

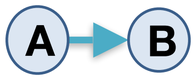
A simple graph G(V,E) is a graph which contains **no multi-edges** and **no loops**

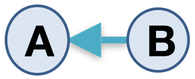


Not a simple graph!

→ A ***general graph***

Graph Term: Directed Graph

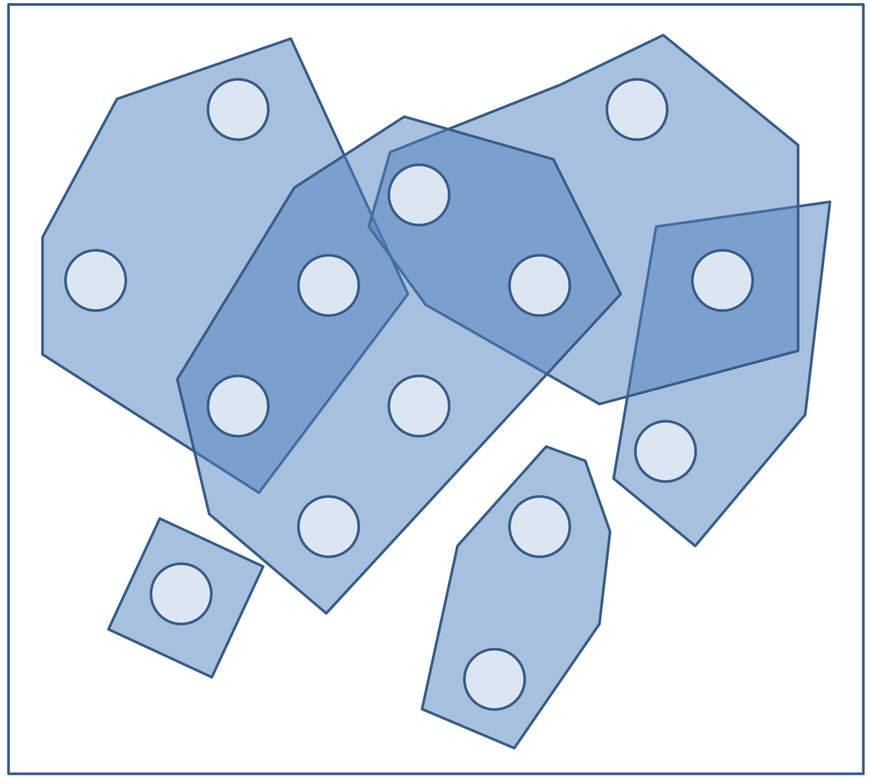
A directed graph (digraph) is a graph that discerns between the edges 

and 

.

Graph Terms: Hypergraph

A hypergraph is a graph with edges connecting any number of vertices.

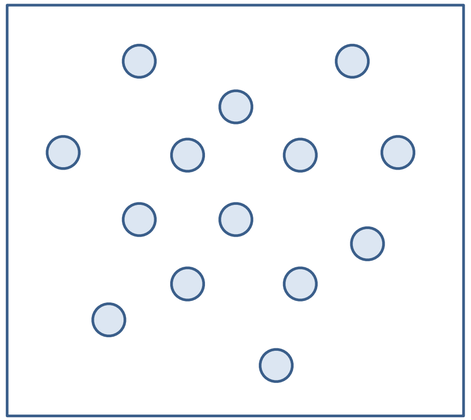


Hypergraph Example

Graph Terms

Independent Set

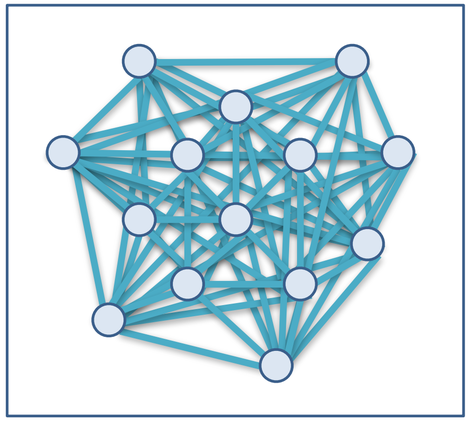
G contains no edges



Independent Set

Clique

G contains all possible edges

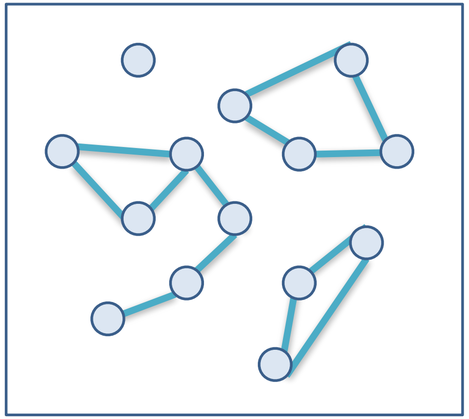


Clique

Unconnected Graphs, Articulation Points

Unconnected graph

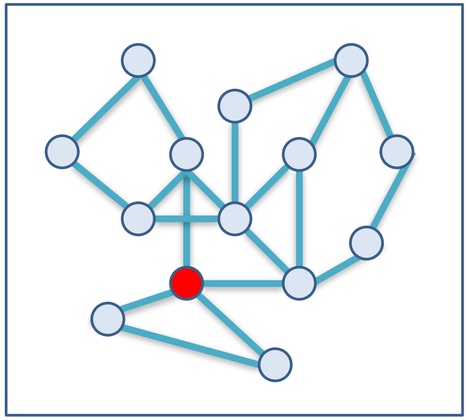
An edge traversal starting from a given vertex cannot reach any other vertex.



Unconnected Graph

Articulation point

Vertices, which if deleted from the graph, would break up the graph in multiple sub-graphs.

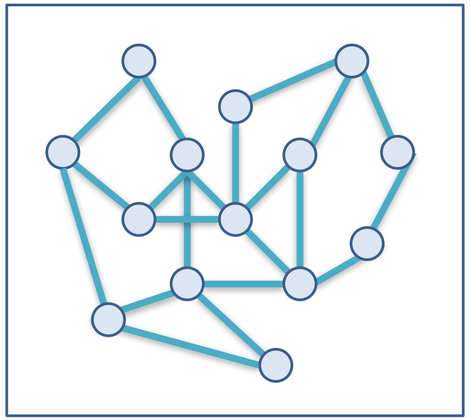


Articulation Point (red)

Biconnected, Bipartite Graphs

Biconnected graph

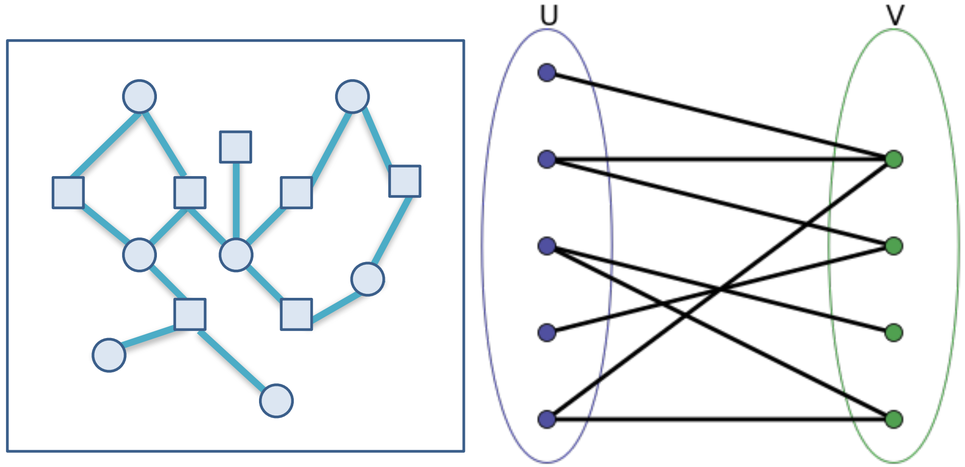
A graph without articulation points.



Biconnected Graph

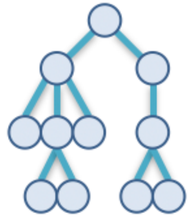
Bipartite graph

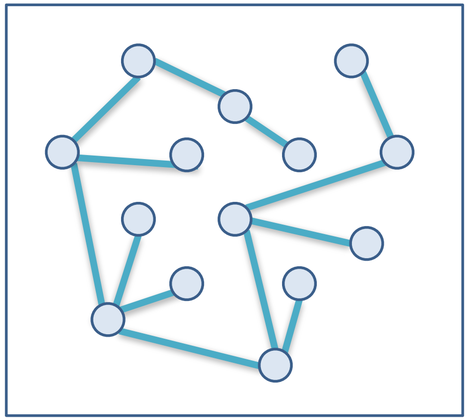
The vertices can be partitioned in two independent sets.



Bipartite Graph

Tree



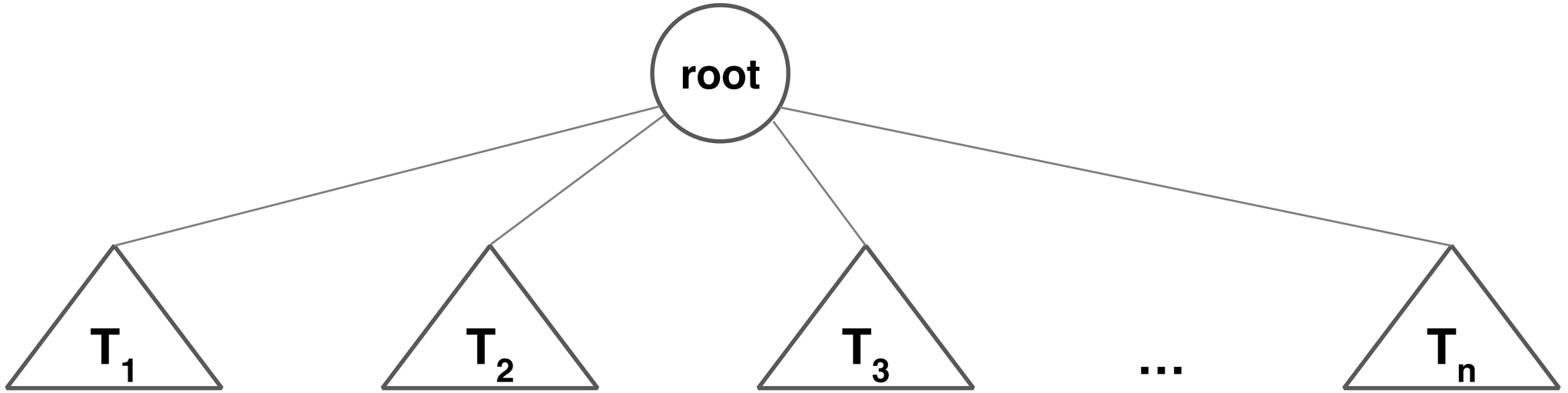


A graph with no cycles - or:

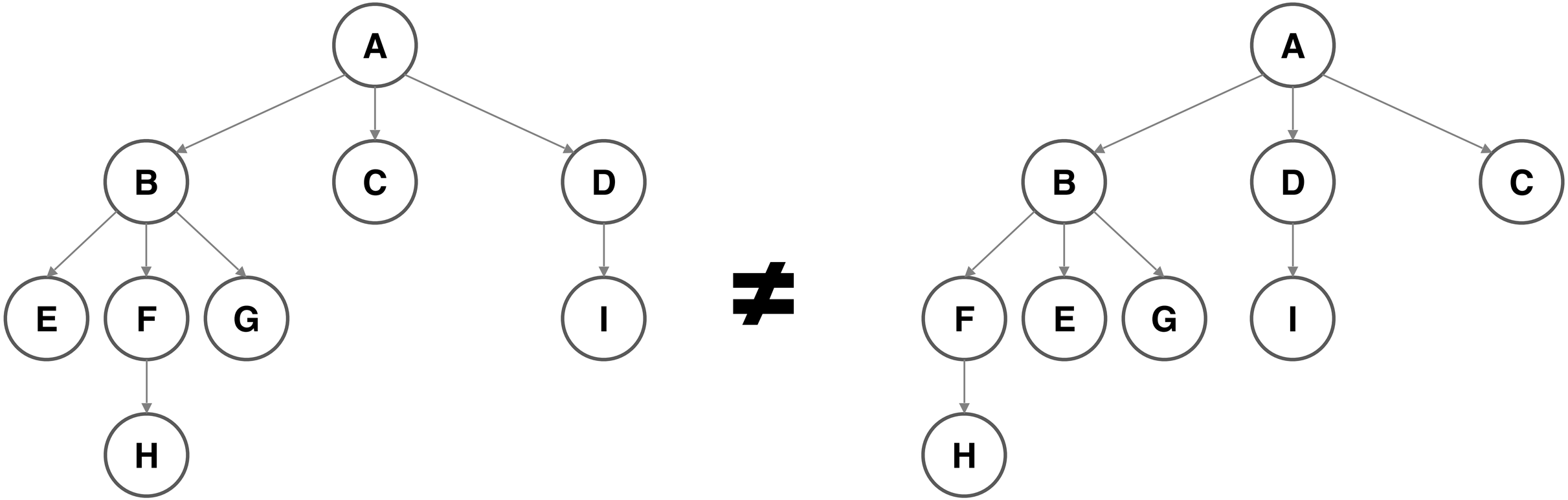
A collection of nodes

contains a root node and 0-n subtrees

subtrees are connected to root by an edge



Ordered Tree



Degree

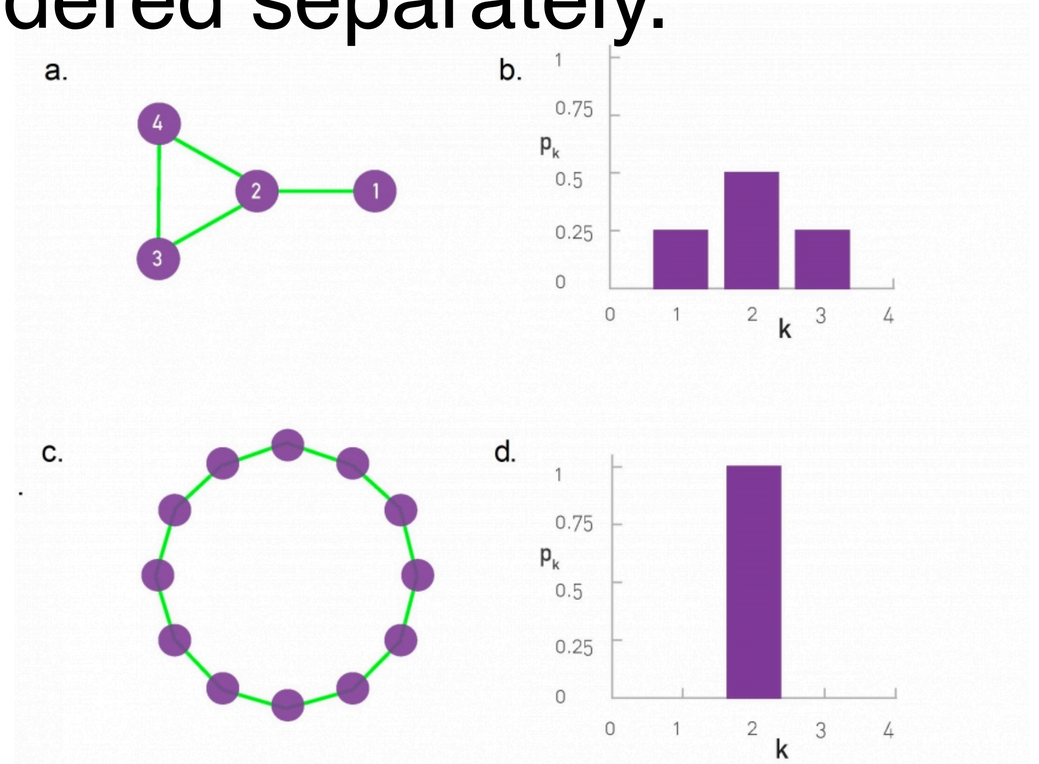
Node degree deg(x)

The number of edges connecting to a node. For directed graphs indeg/outdeg are considered separately.

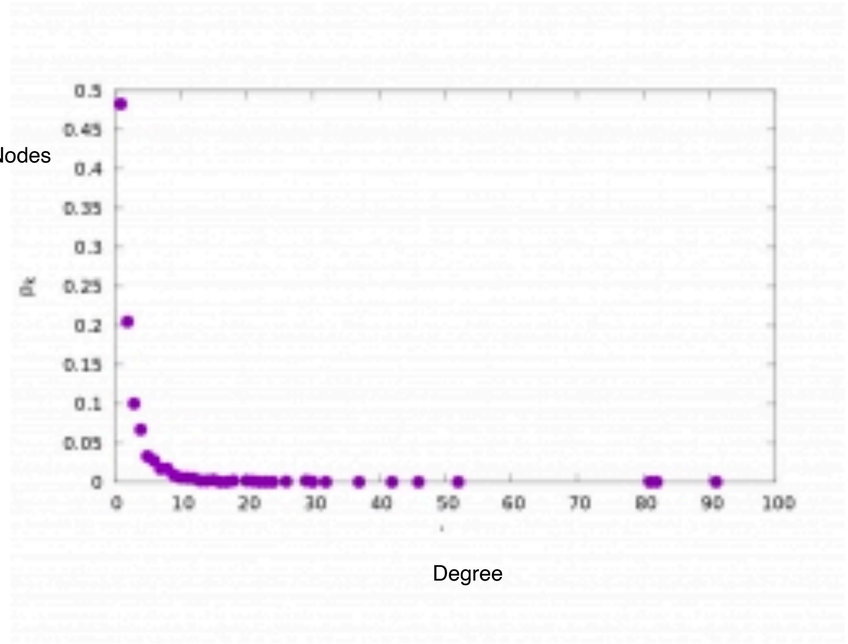
Average degree

$$\langle k \rangle = \frac{1}{N} \sum_{i=1}^{N} k_i = \frac{2L}{N}$$

Degree distribution



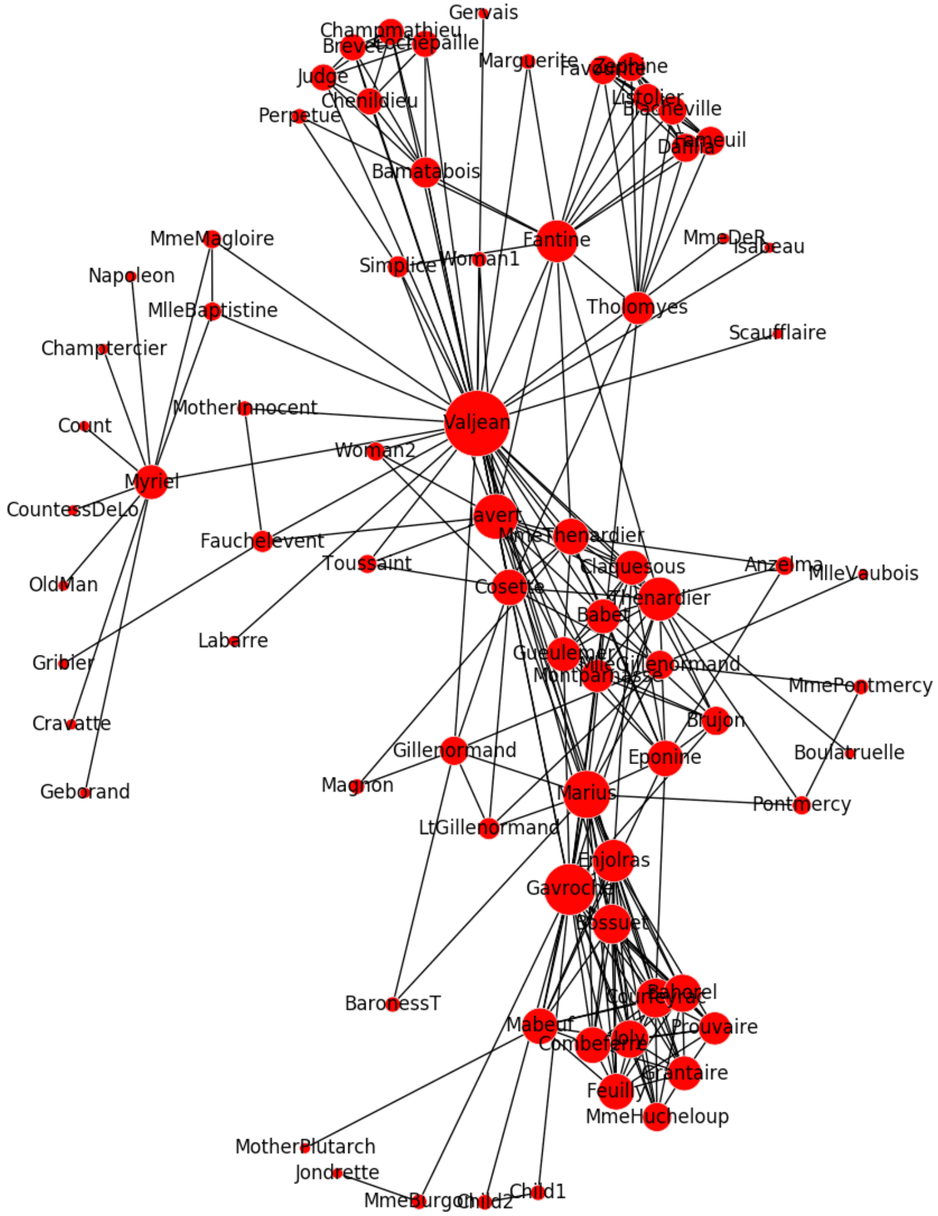
Degree Distribution of a real Network



Protein Interaction Network

Degrees

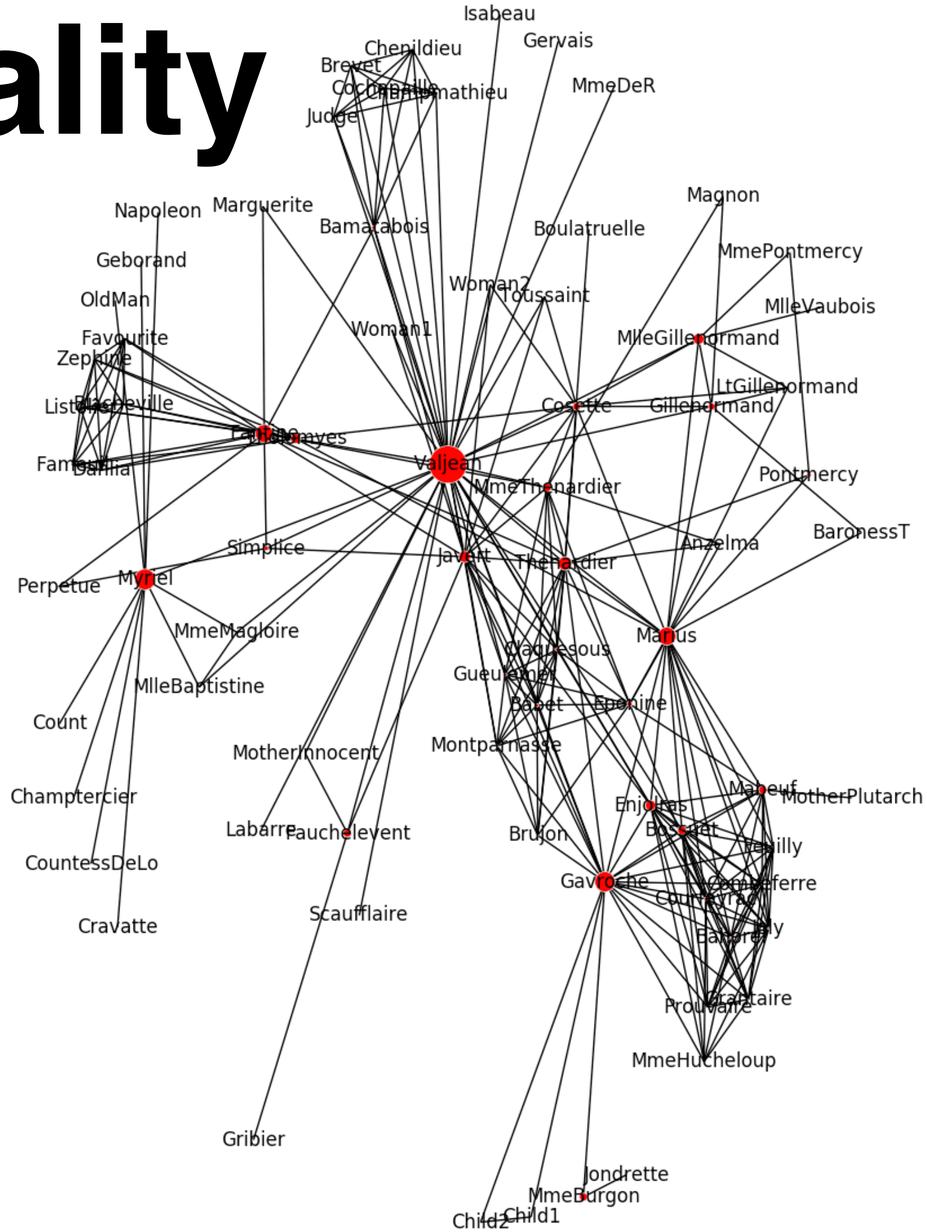
Degree is a measure of local importance



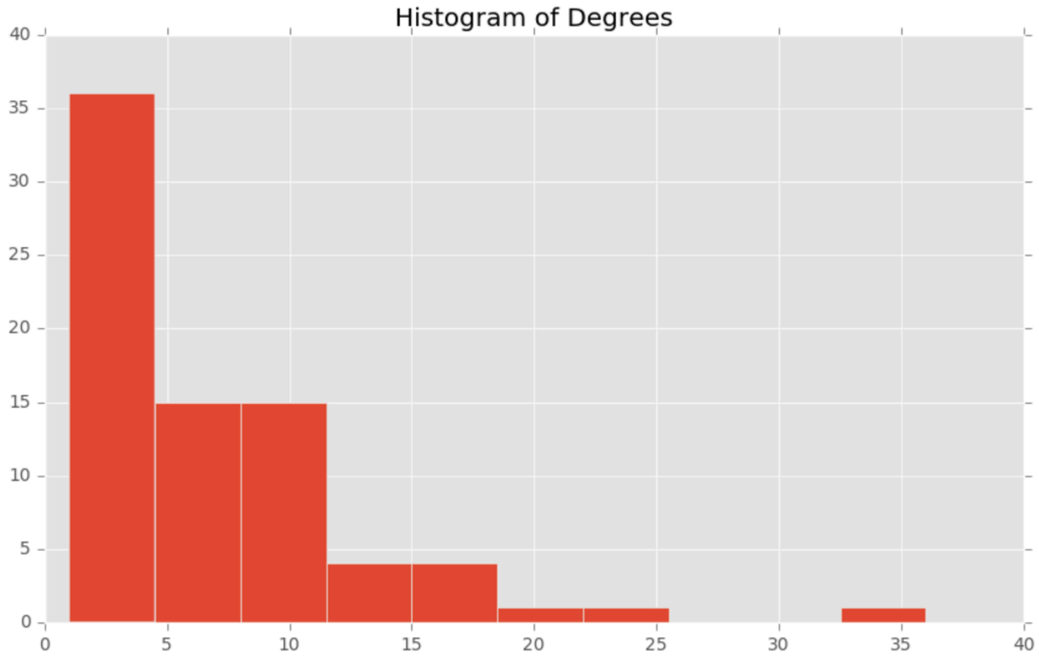
Betweenness Centrality

a measure of how many shortest paths pass through a node

good measure for the overall relevance of a node in a graph



Degree vs BC


Histogram of Degrees
40
35
30
25
20
15
10
5
0
0
5
10
15
20
25
30
35
40


Histogram of Betweennes Centrality Values
80
70
60
50
40
30
20
10
0
0.0
0.1
0.2
0.3
0.4
0.5
0.6

Adjacency Matrix

Represent Nodes as rows and columns of a matrix

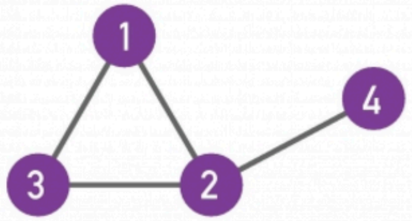
Cells != 0 if there is a link

Degree of node is sum of its column

$A_{ij} = 1$ A link pointing from node *j* to node *i*

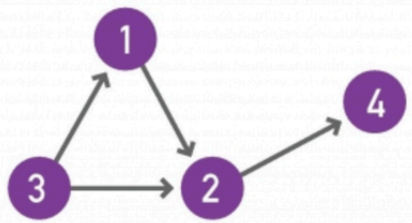
$$A_{ij} = \begin{matrix} A_{11} & A_{12} & A_{13} & A_{14} \\ A_{21} & A_{22} & A_{23} & A_{24} \\ A_{31} & A_{32} & A_{33} & A_{34} \\ A_{41} & A_{42} & A_{43} & A_{44} \end{matrix}$$

b. **Undirected network**



$$A_{ij} = \begin{matrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{matrix}$$

c. **Directed network**



$$A_{ij} = \begin{matrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{matrix}$$

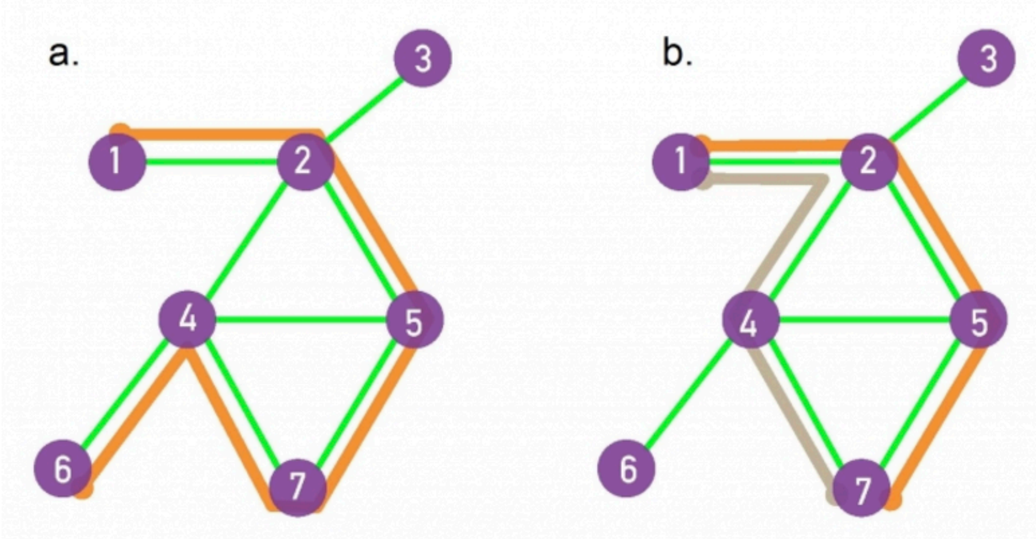
Paths & Distances

Path is **route along links**

Path length is the **number of links** contained

Shortest paths connects nodes i and j with the smallest number of links

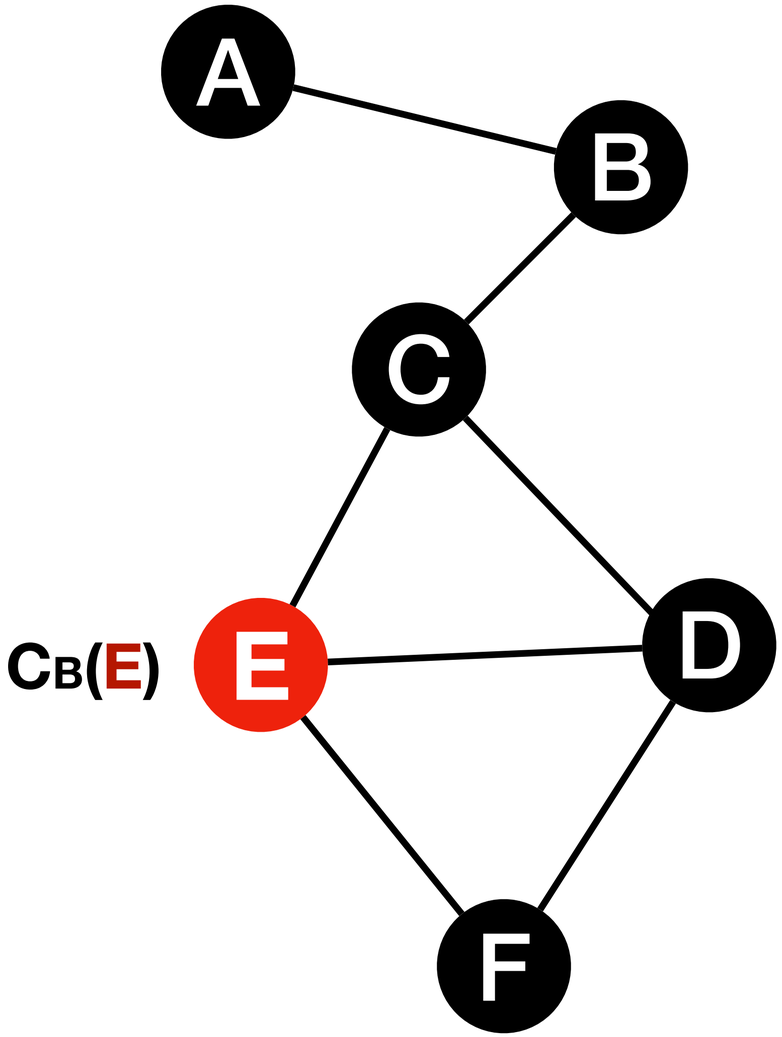
Diameter of graph G
The longest shortest path within G.



A path from 1 to 6

Shortest paths (two) from 1 to 7.

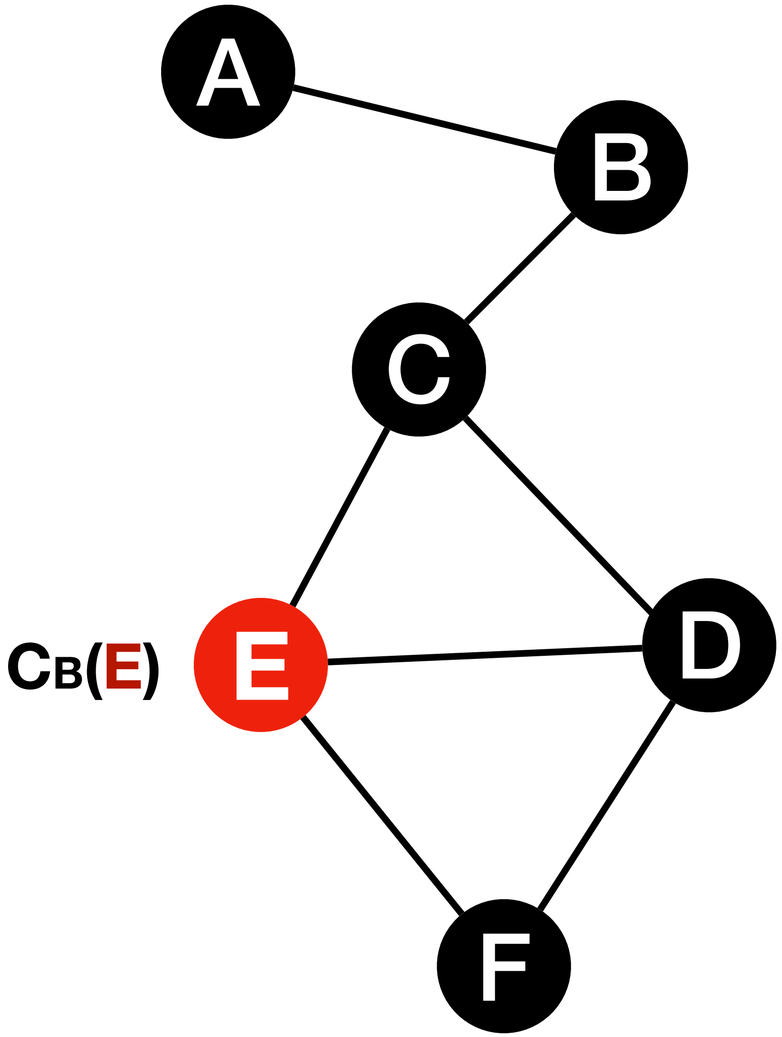
Betweenness Centrality $c_B(v) = \sum_{s \neq v \neq t \in V} \frac{\sigma(s,t|v)}{\sigma(s,t)}$



Shortest Path

A, B
A, C
A, D
A, F
B, C
B, D
B, F
C, D
C, F
D, F

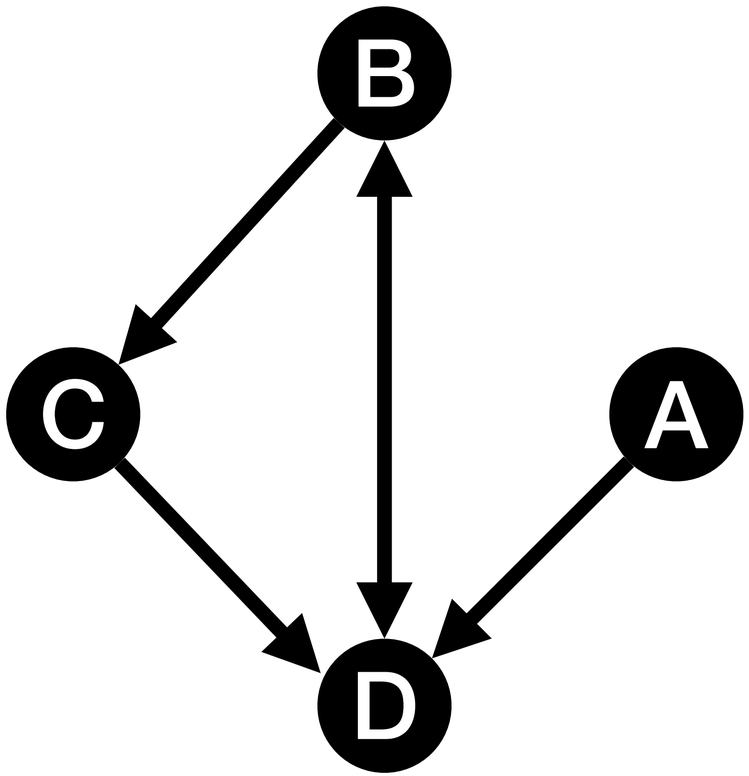
Betweenness Centrality $c_B(v) = \sum_{s \neq v \neq t \in V} \frac{\sigma(s,t|v)}{\sigma(s,t)}$



Shortest Path

A, B
A, C
A, D
A, F
B, C
B, D
B, F
C, D
C, F
D, F

Page Rank



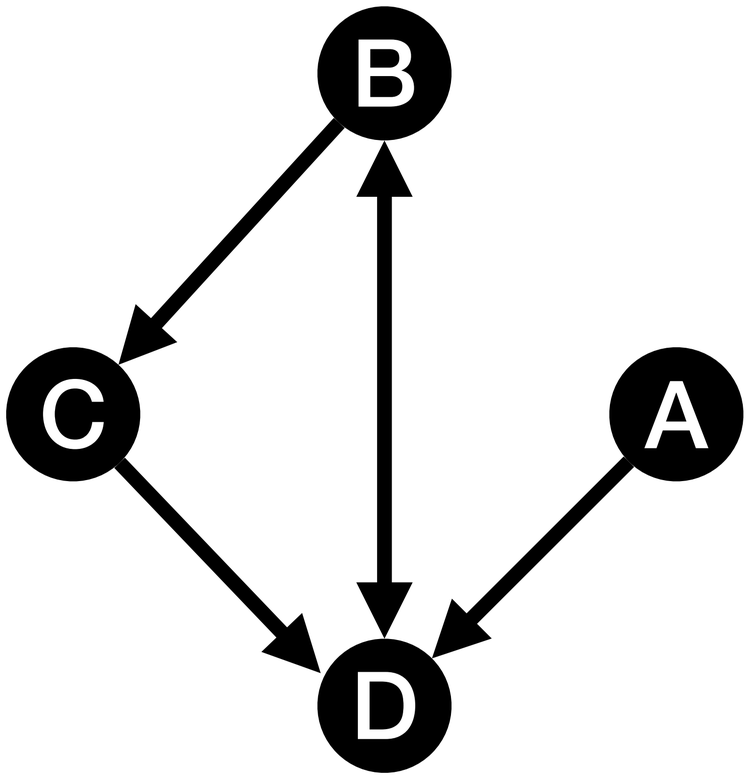
A

B

C

D

Page Rank



A

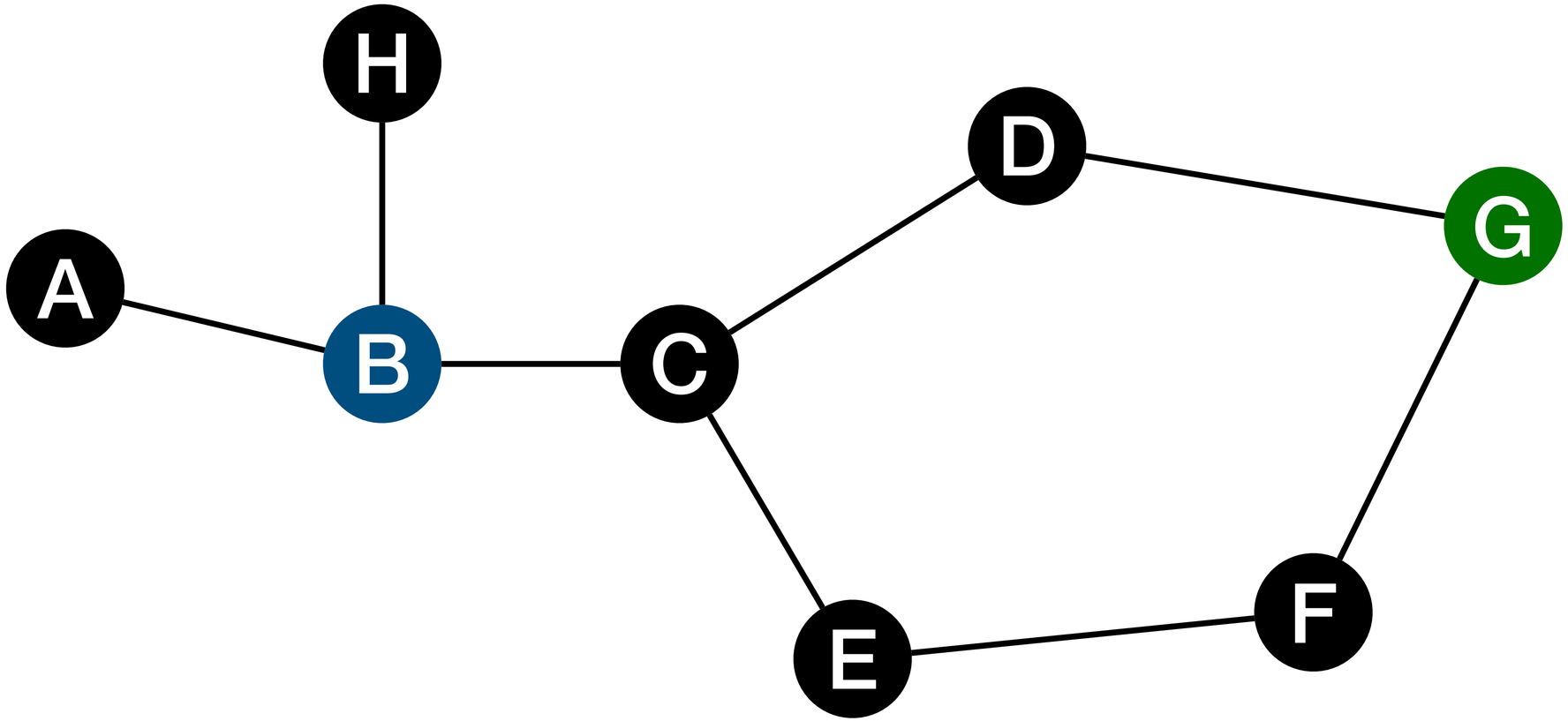
B

C

D

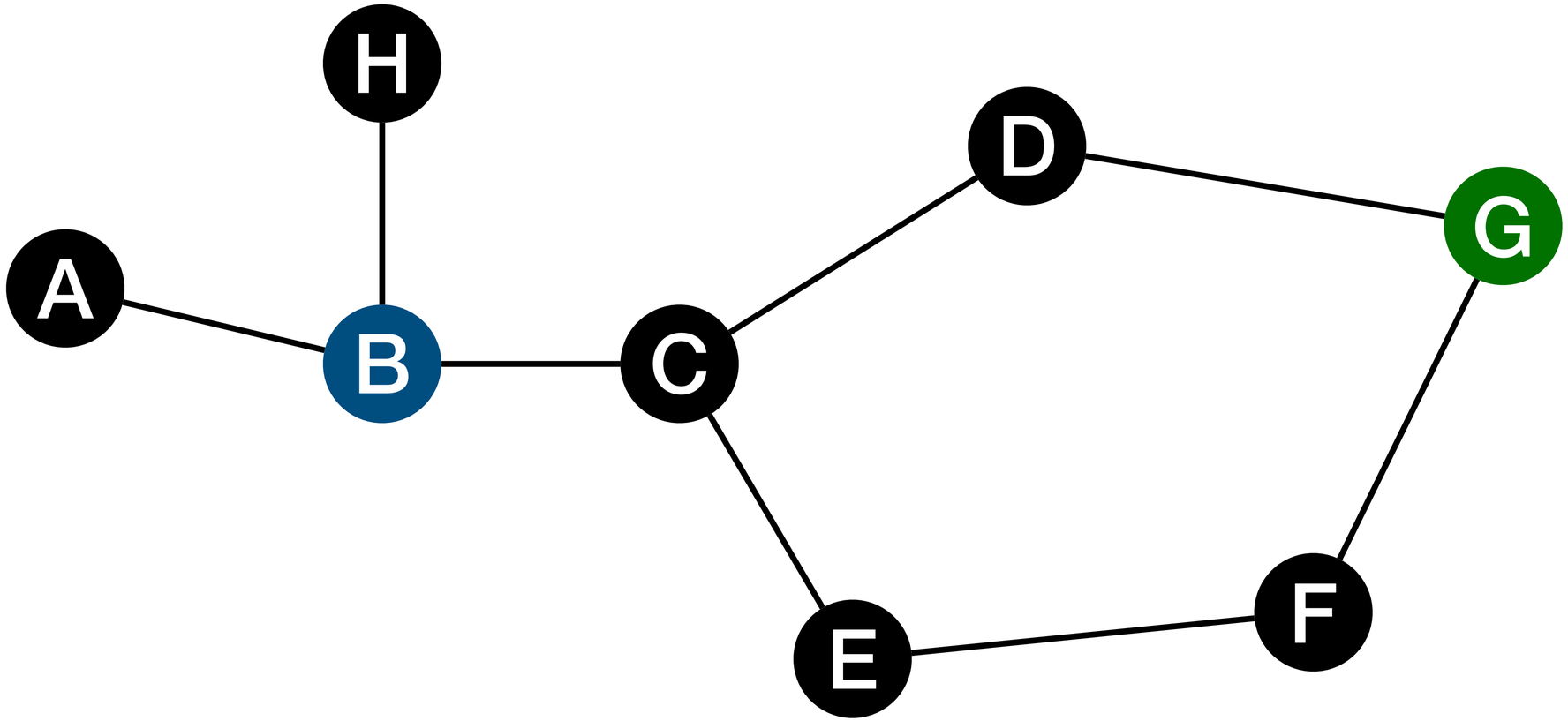
Breadth First Search

Shortest Path: (B, G)



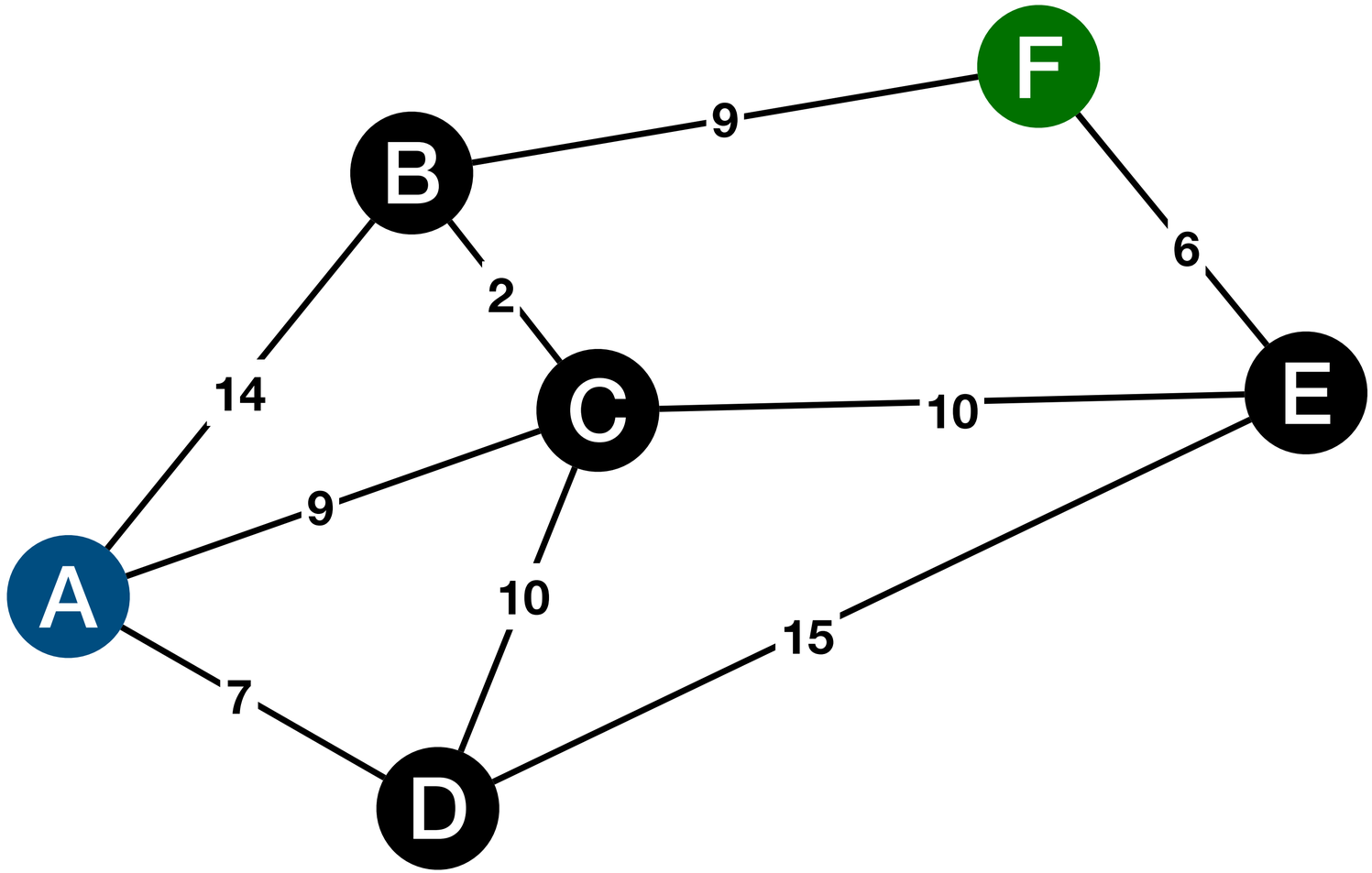
Breadth First Search

Shortest Path: (B, G)



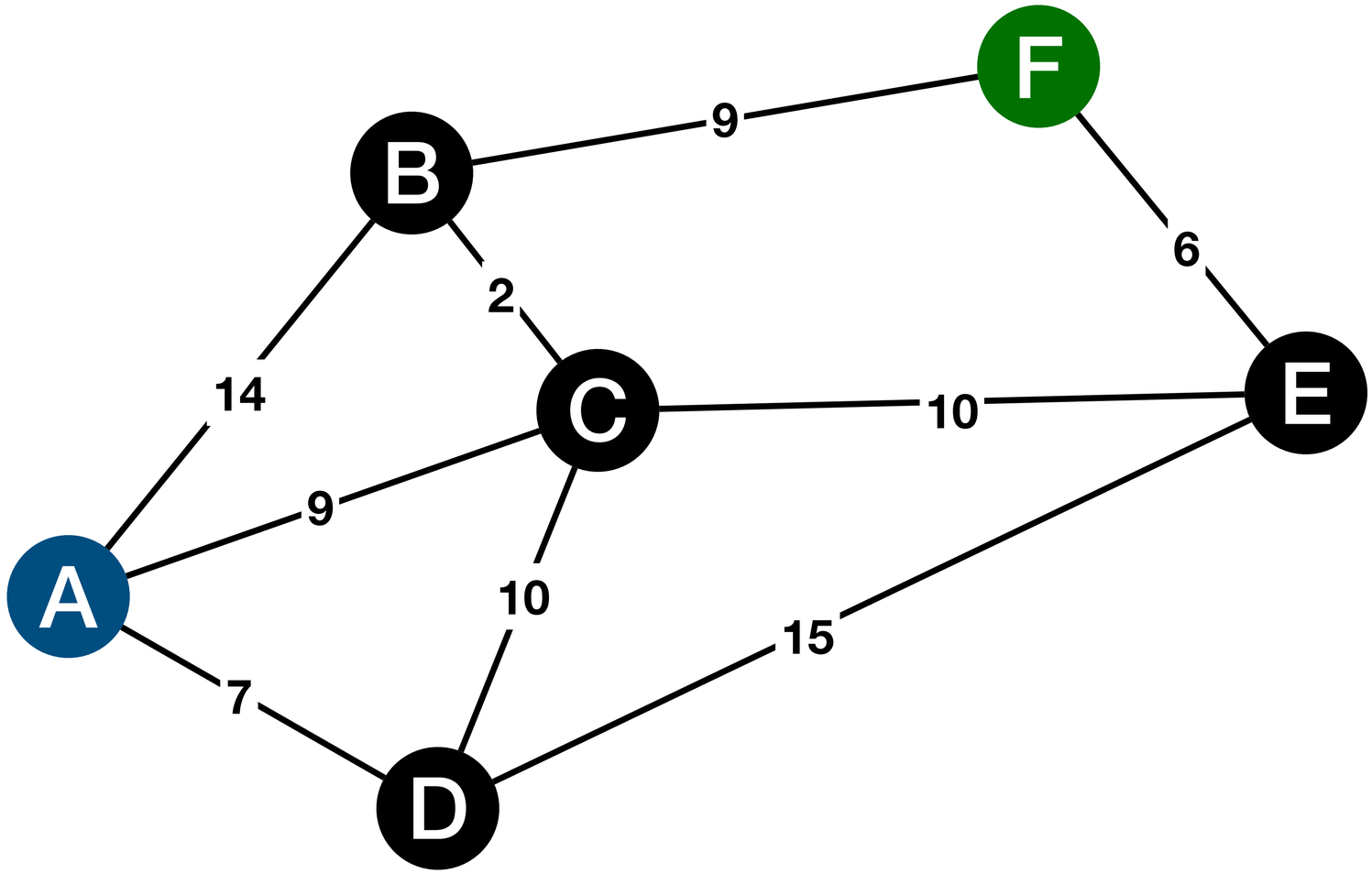
Dijkstra's Algorithm

Shortest Path: (A, F)



Dijkstra's Algorithm

Shortest Path: (A, F)



Graph and Tree Visualization

Different Kinds of Tasks/Goals

Two principal types of tasks: **attribute-based (ABT)** and **topology-based (TBT)**

Localize – find a single or multiple nodes/edges that fulfill a given property

- ABT: Find the edge(s) with the maximum edge weight.
- TBT: Find all adjacent nodes of a given node.

Quantify – count or estimate a numerical property of the graph

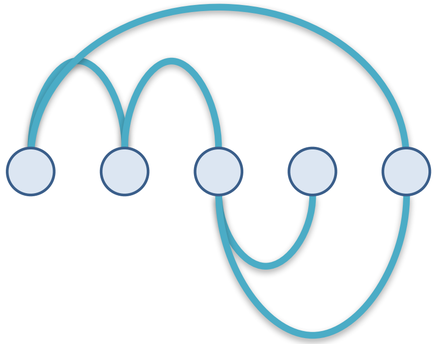
- ABT: Give the number of all nodes.
- TBT: Give the degree of a node.

Sort/Order – enumerate the nodes/edges according to a given criterion

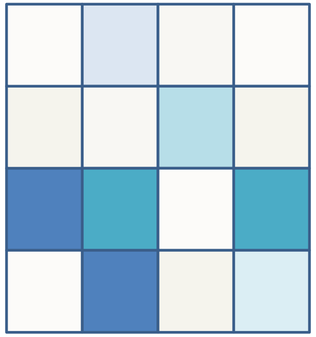
- ABT: Sort all edges according to their weight.
- TBT: Traverse the graph starting from a given node.

list adapted from Schulz 2010

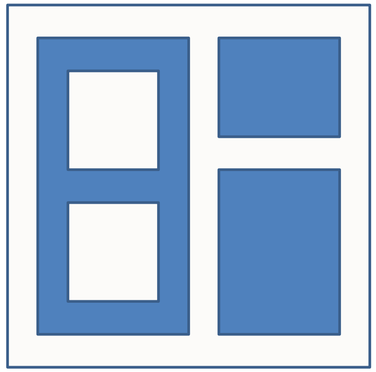
Three Types of Graph Representations



Explicit
(Node-Link)



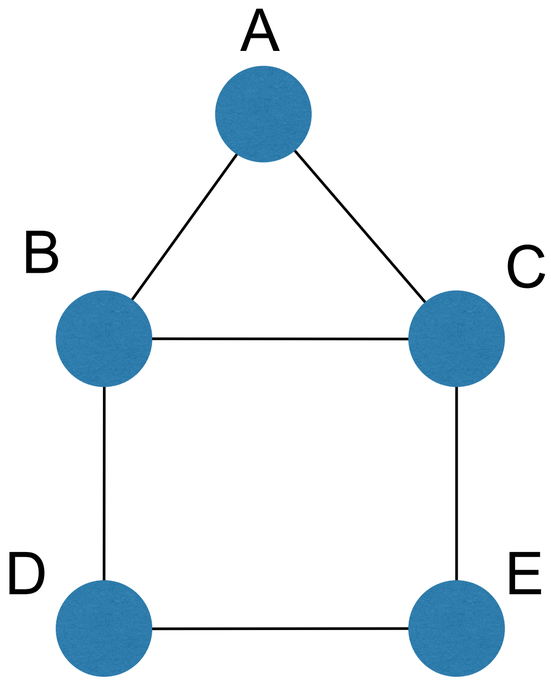
Matrix



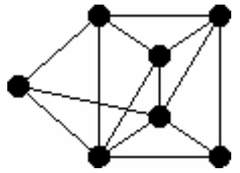
Implicit

Explicit Graph Representations

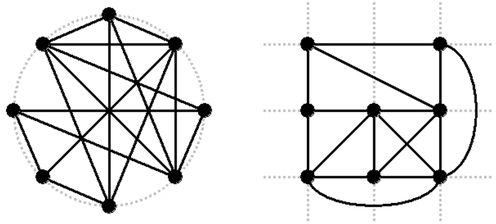
Node-link diagrams: vertex = point, edge = line/arc



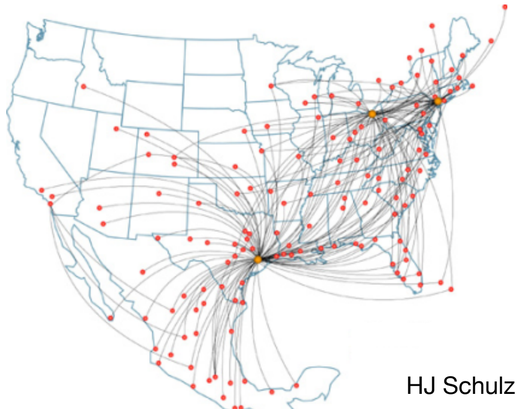
Free



Styled



Fixed



HJ Schulz 2006

Criteria for Good Node-Link Layout

Minimized **edge crossings**

Minimized **distance** of neighboring nodes

Minimized **drawing area**

Uniform edge **length**

Minimized edge **bends**

Maximized **angular distance** between different edges

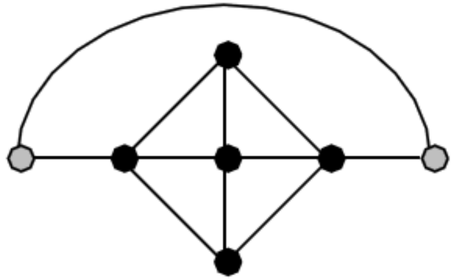
Aspect ratio about 1 (not too long and not too wide)

Symmetry: similar graph structures should look similar

list adapted from Battista et al. 1999

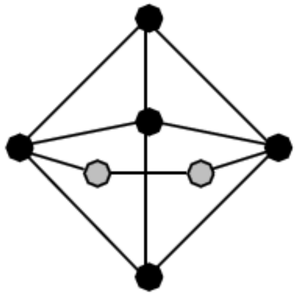
Conflicting Criteria

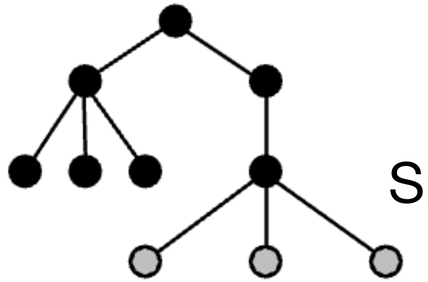
Minimum number of edge crossings



vs.

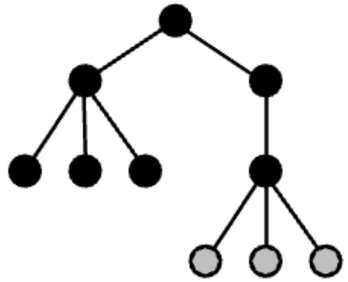
Uniform edge length





Space utilization

vs.



Symmetry

Schulz 2004

Force Directed Layouts

Physics model:
edges = springs,
vertices = repulsive magnets

Expander
(pushing nodes apart)

Spring Coil
(pulling nodes together)

Algorithm

Place Vertices in random locations

While not equilibrium

  calculate force on vertex

    sum of

      pairwise repulsion of all nodes

      attraction between connected nodes

  move vertex by c * force on vertex

Properties

Generally good layout

Uniform edge length

Clusters commonly visible

Not deterministic

Computationally expensive: $O(n^3)$

n^2 in every step, it takes about n cycles to reach equilibrium

Limit (interactive): ~1000 nodes

in practice: damping, center of gravity

http://bl.ocks.org/steveharoz/8c3e2524079a8c440df60c1ab72b5d03

Giant Hairball

Graphs in 3D

Why, why not visualize graphs in 3D?

Why, why not use AR/VR?

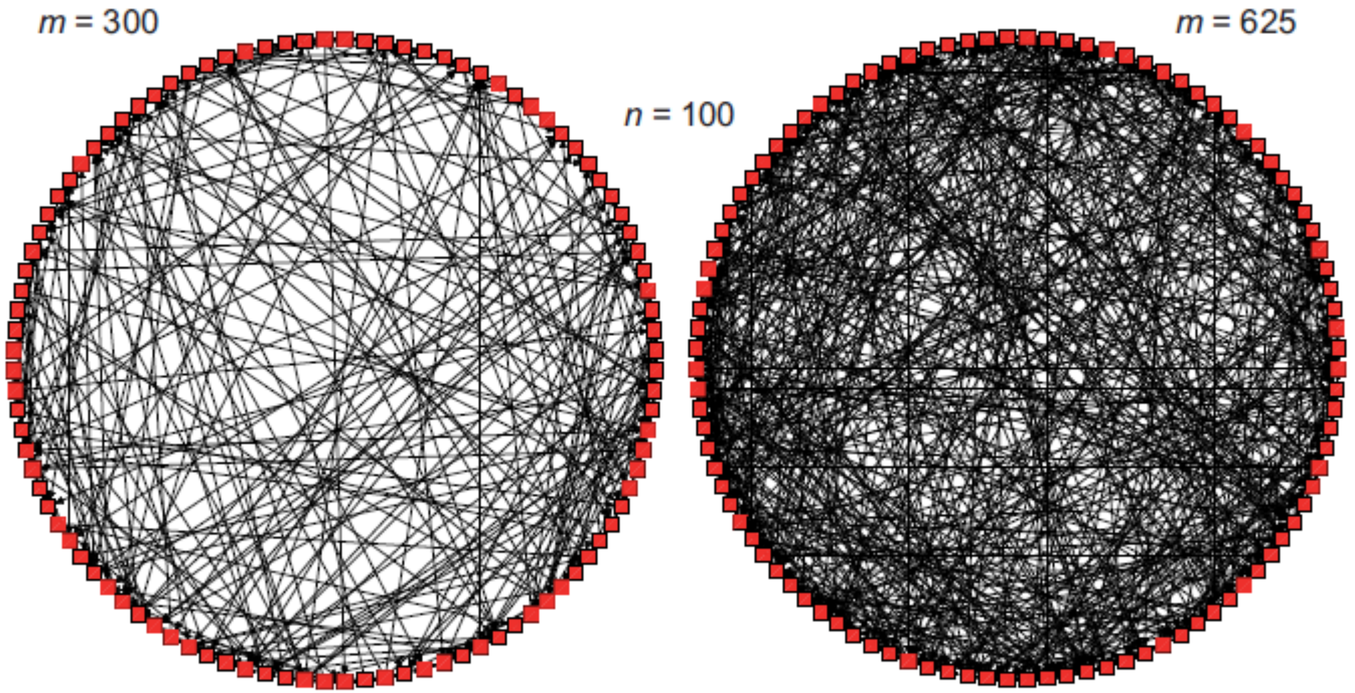
https://twitter.com/alexsigaras/status/860560655031685121

Styled / Restricted Layouts

Circular Layout

Node ordering

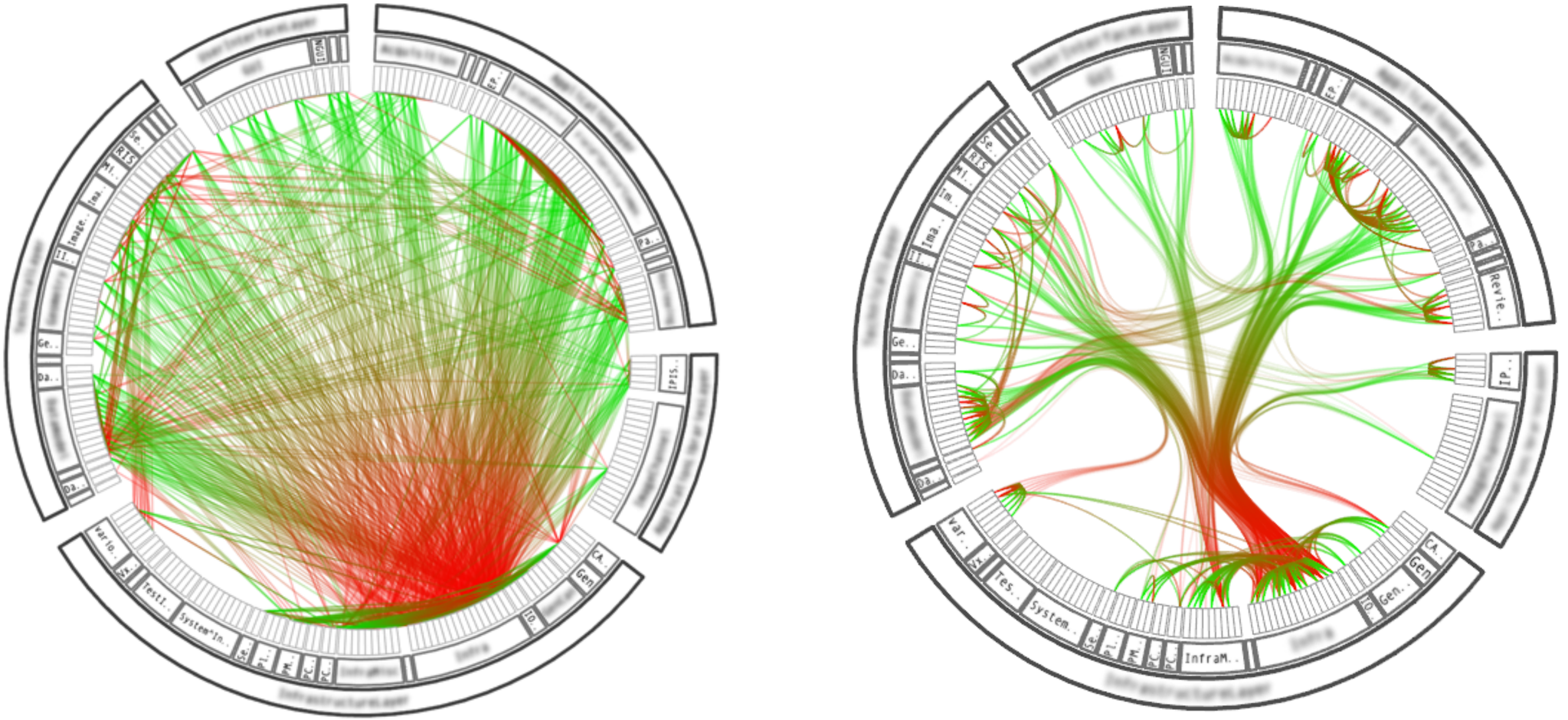
Edge Clutter



ca. 3% of all possible edges

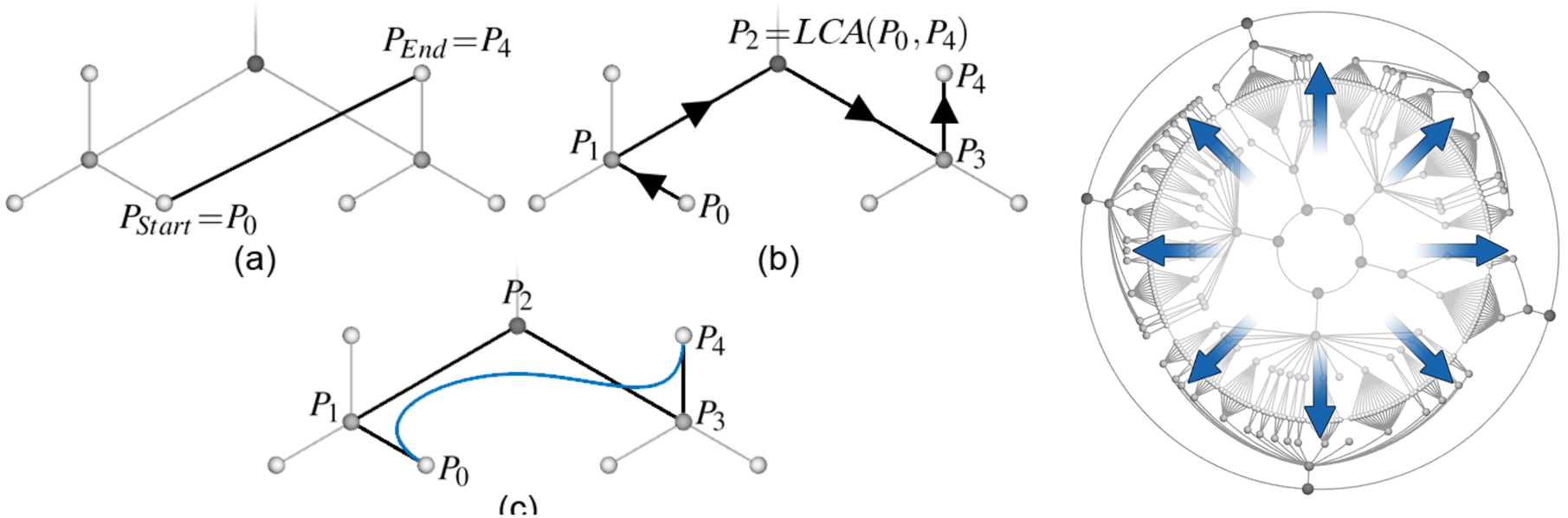
ca. 6,3% of all possible edges

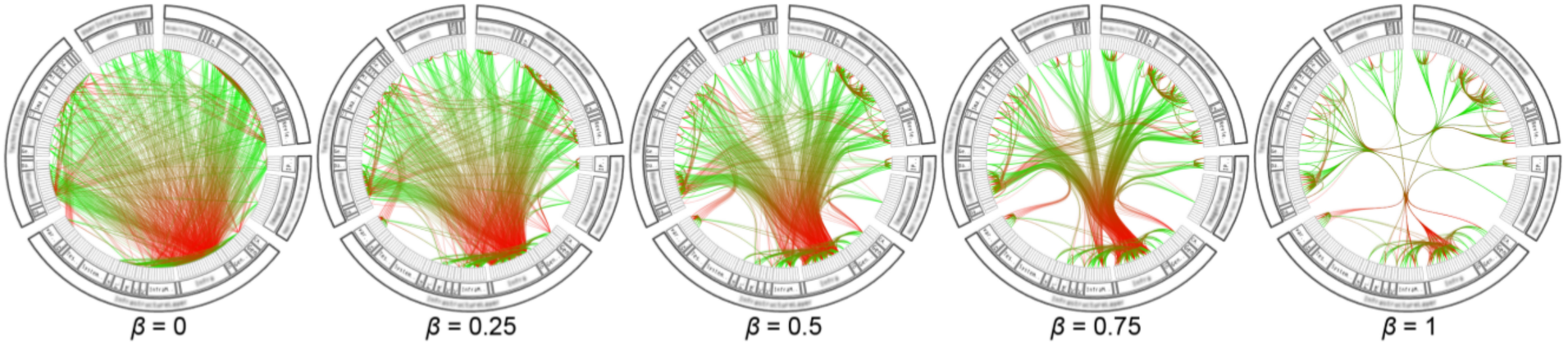
Reduce Clutter: Edge Bundling



Holten et al. 2006

Hierarchical Edge Bundling

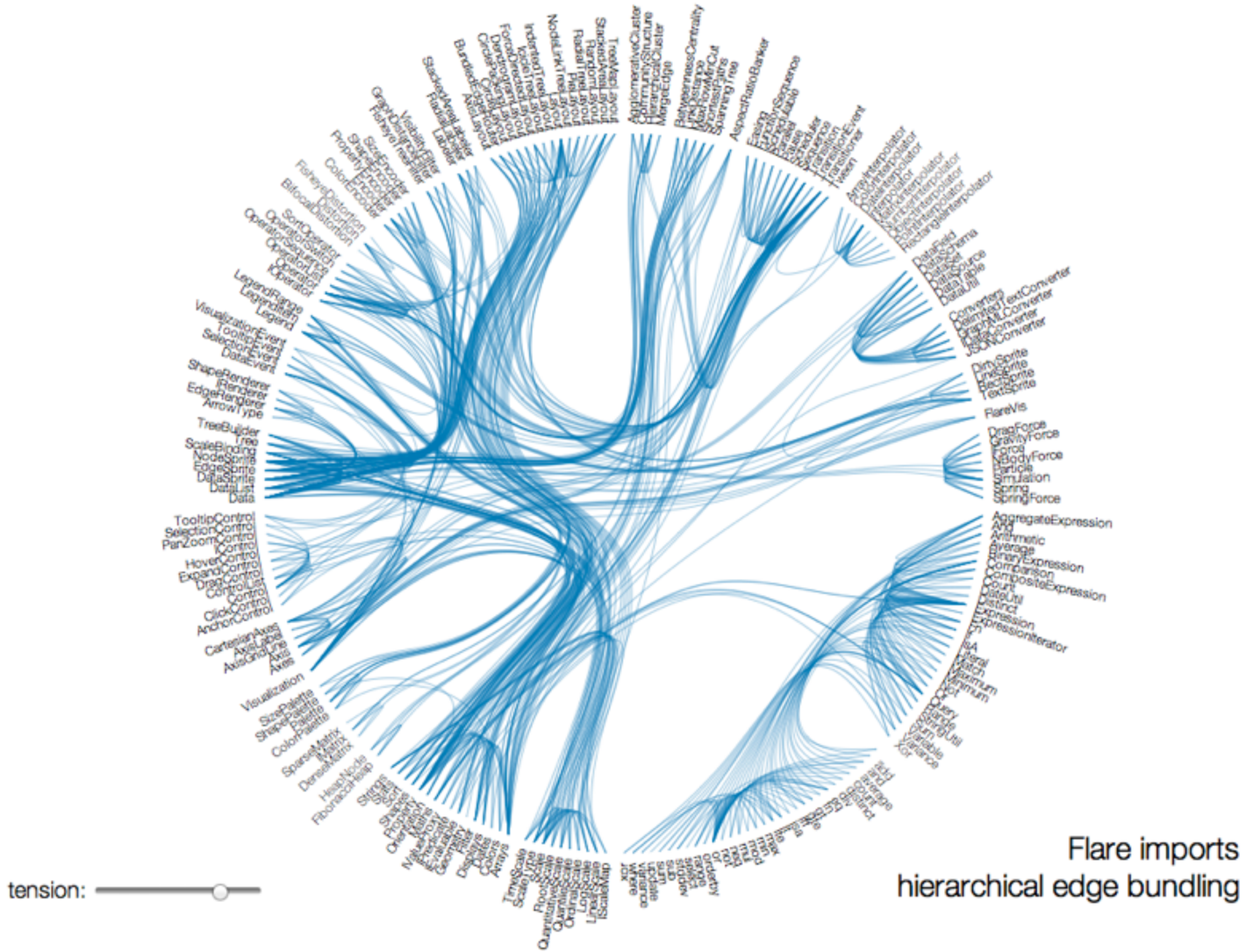




Bundling Strength

Holten et al. 2006

Bundling Strength



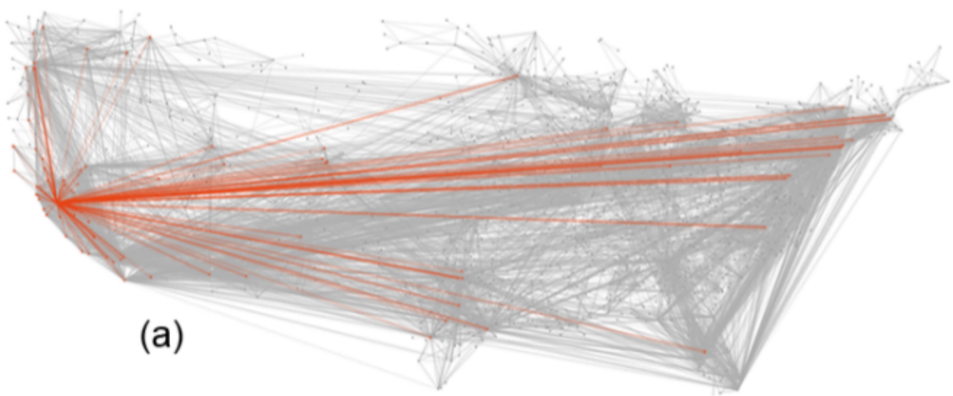
mbostock.github.com/d3/talk/20111116/bundle.html

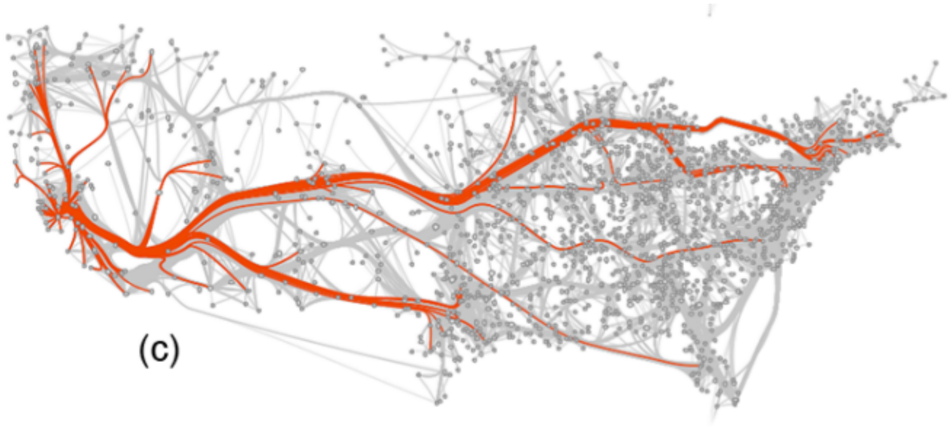
Michael Bostock

Fixed Layouts

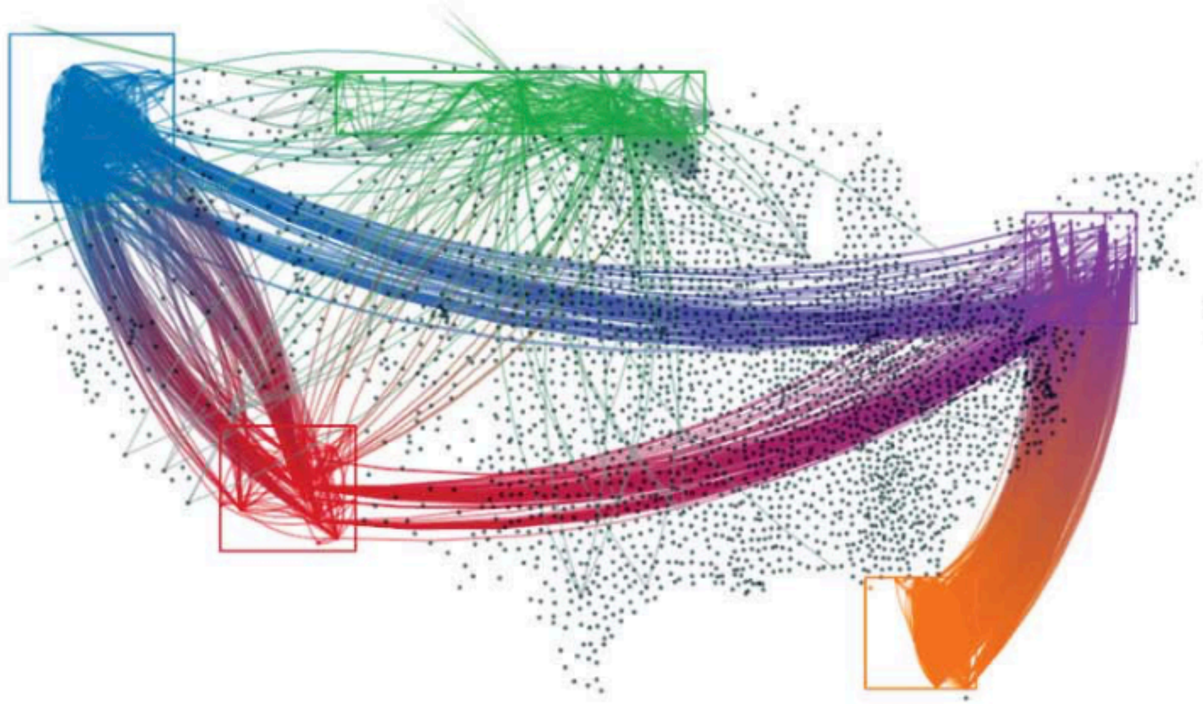
Can’t vary position of nodes

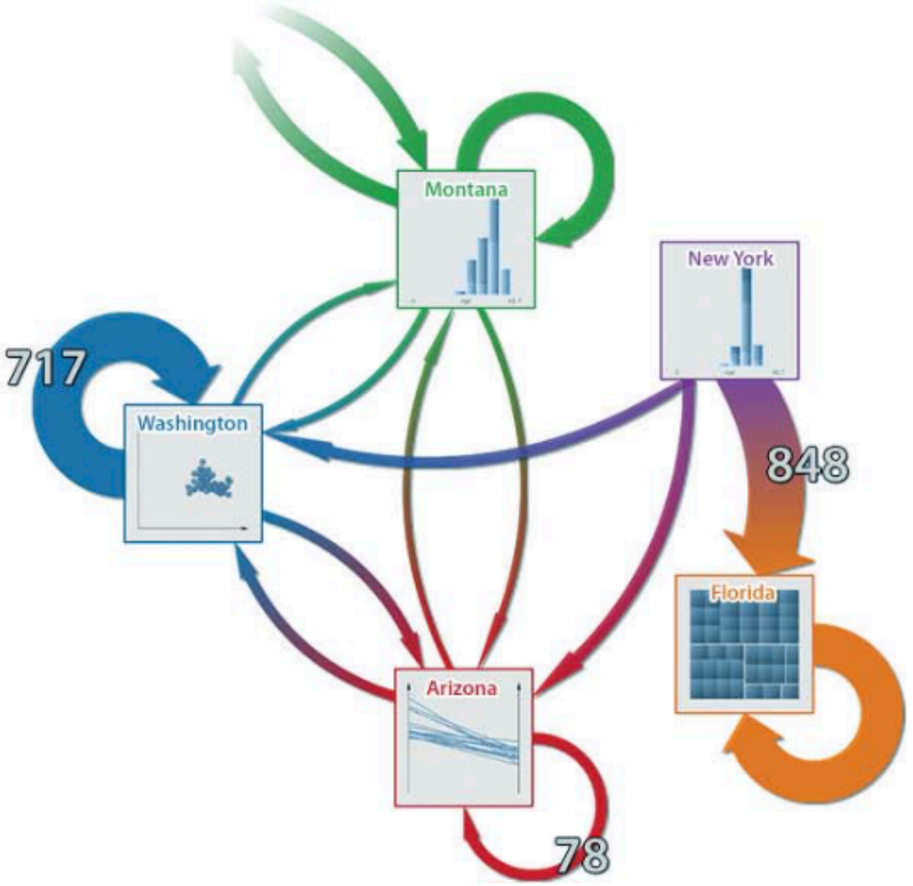
Edge routing important





Aggregation



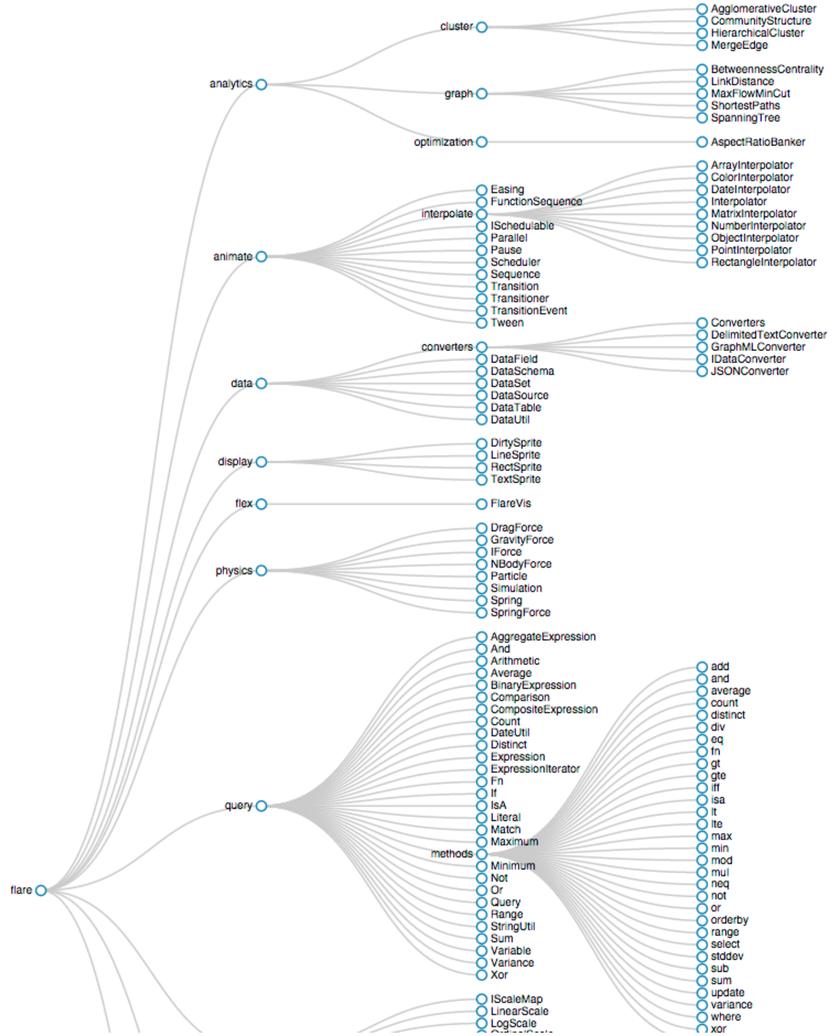


https://www.youtube.com/watch?v=E1PVTitj7h0

Explicit Tree Visualization

Reingold–
Tilford layout

http://billmill.org/pymag-trees/



Explicit Representations

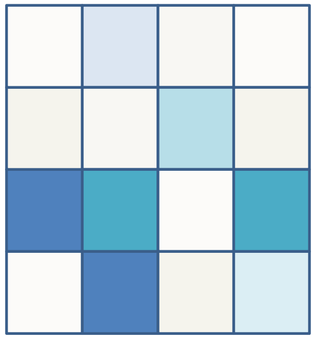
Pros:

- is able to depict all graph classes
- can be customized by weighing the layout constraints
- very well suited for TBTs, if also a suitable layout is chosen

Cons:

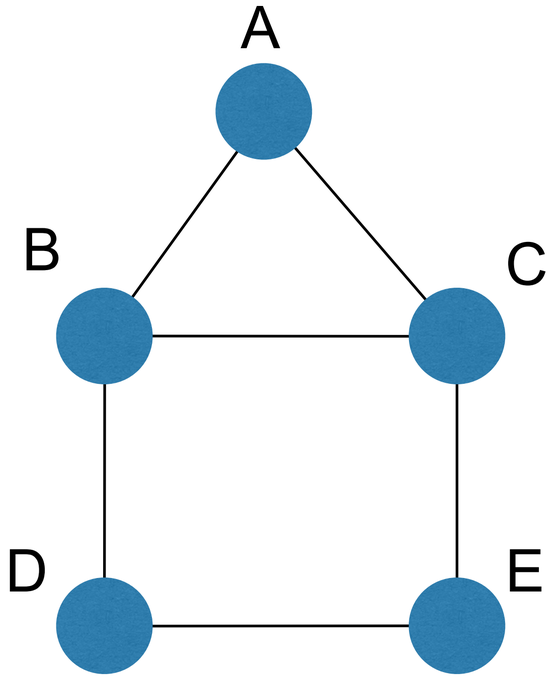
- computation of an optimal graph layout is in NP
  (even just achieving minimal edge crossings is already in NP)
- even heuristics are still slow/complex (e.g., naïve spring embedder is in O(n3))
- has a tendency to clutter (edge clutter, “hairball”)

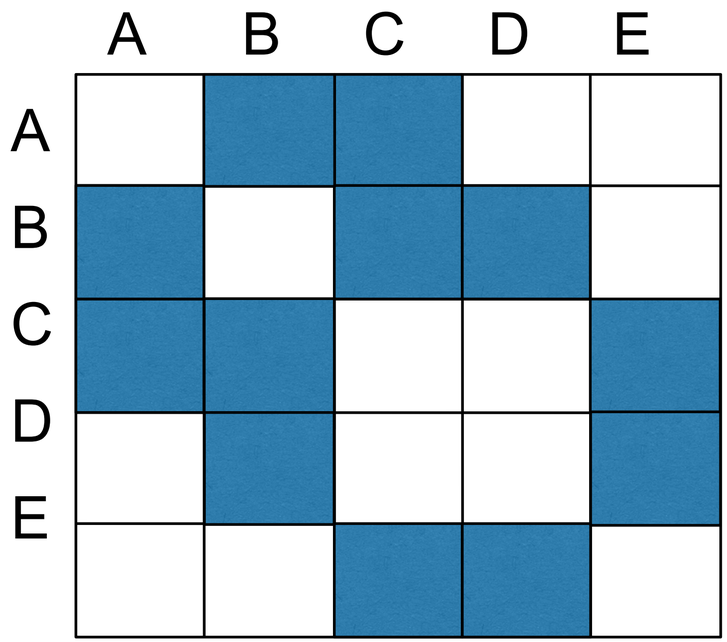
Matrix Representations



Matrix Representations

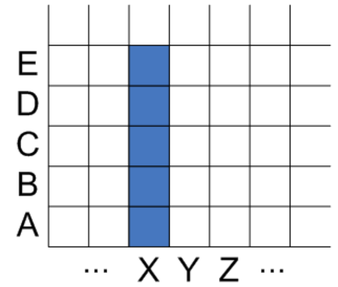
Instead of node link diagram, use adjacency matrix

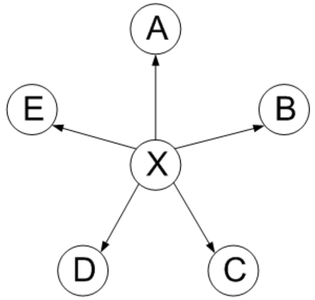


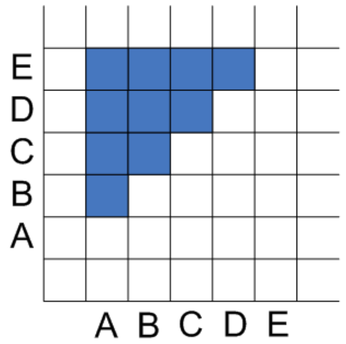


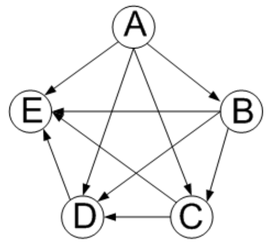
Matrix Representations

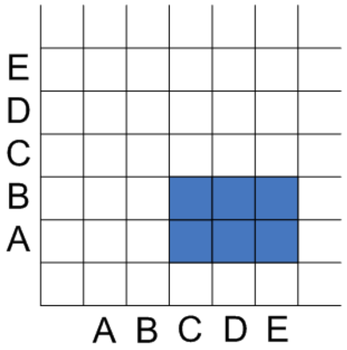
Examples:

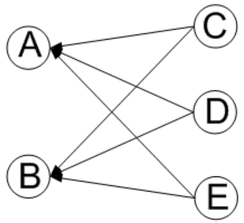


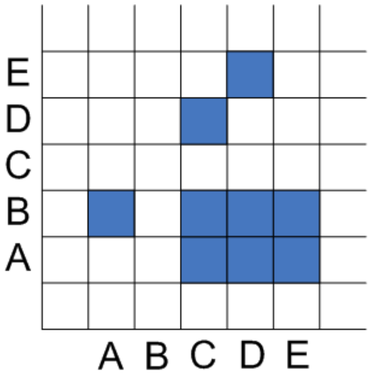


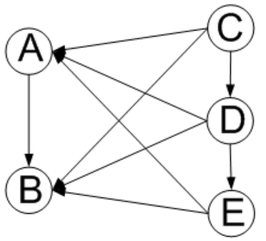




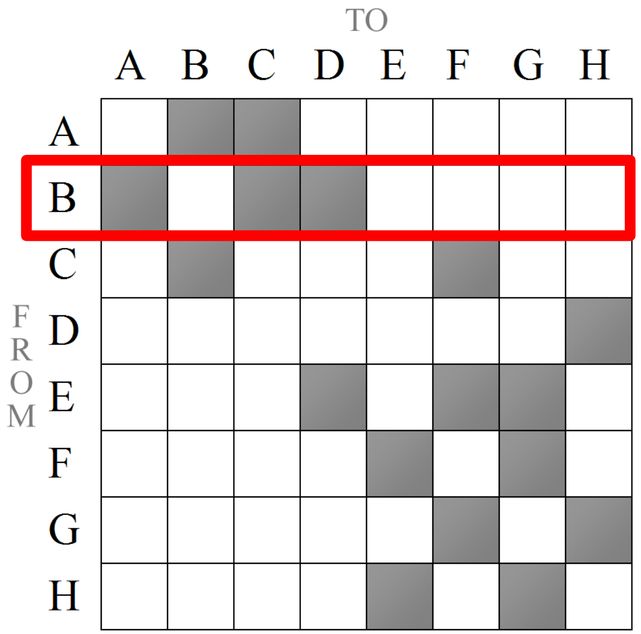




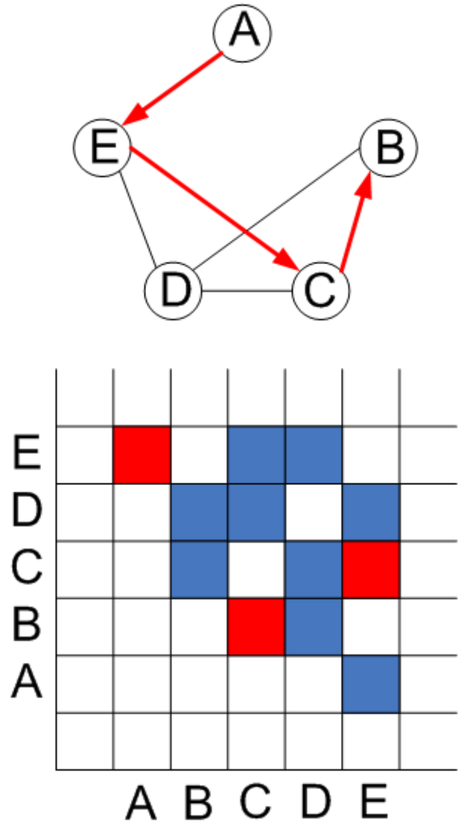




Matrix Representations



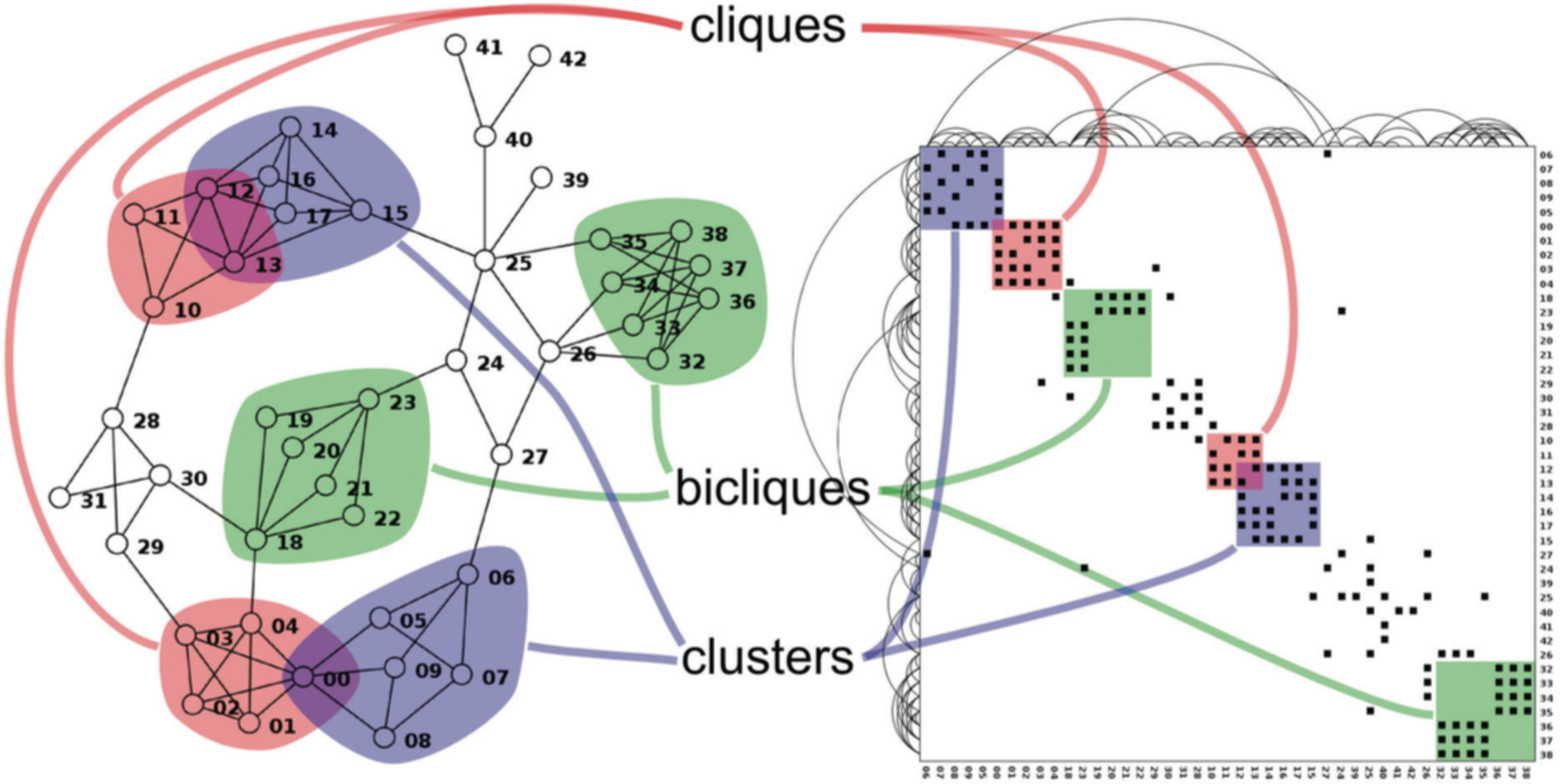
Well suited for
neighborhood-related TBTs



Not suited for
path-related TBTs

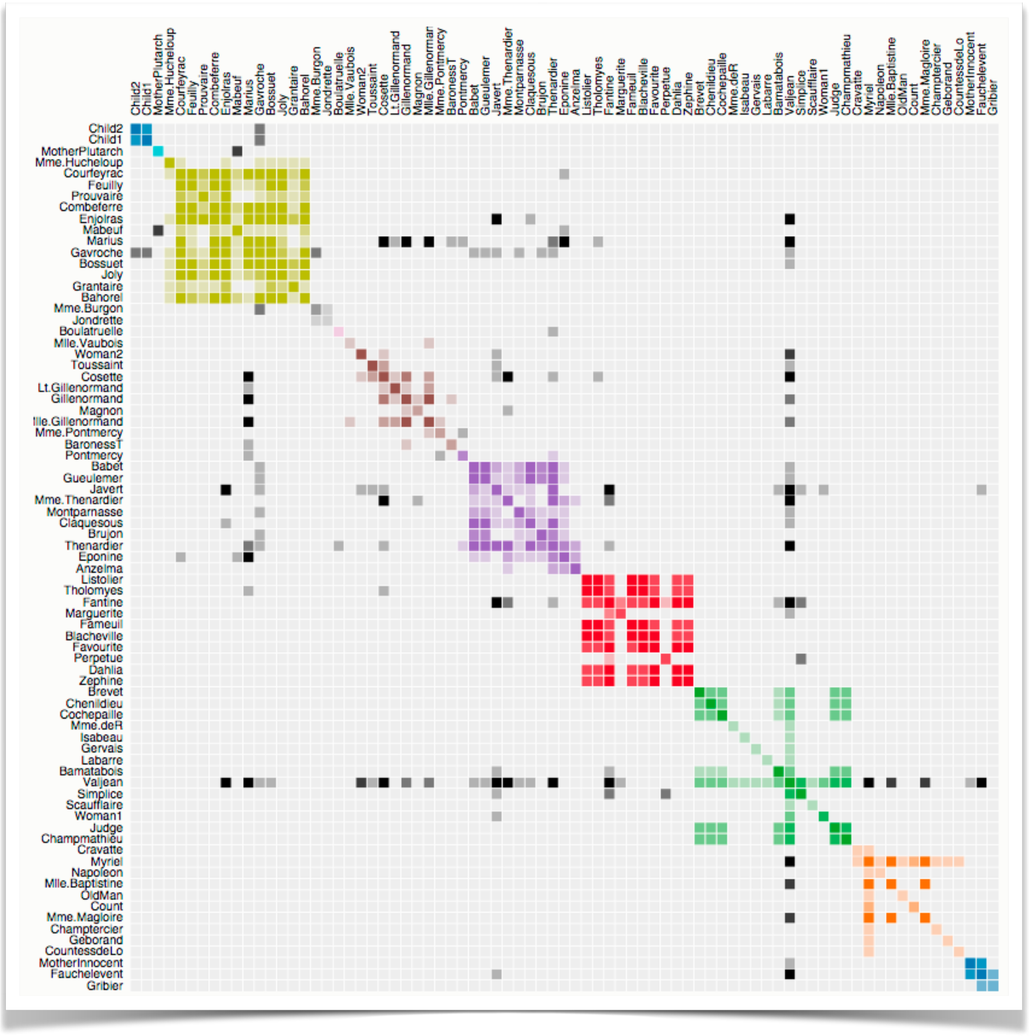
van Ham et al. 2009

Shen et al. 2007

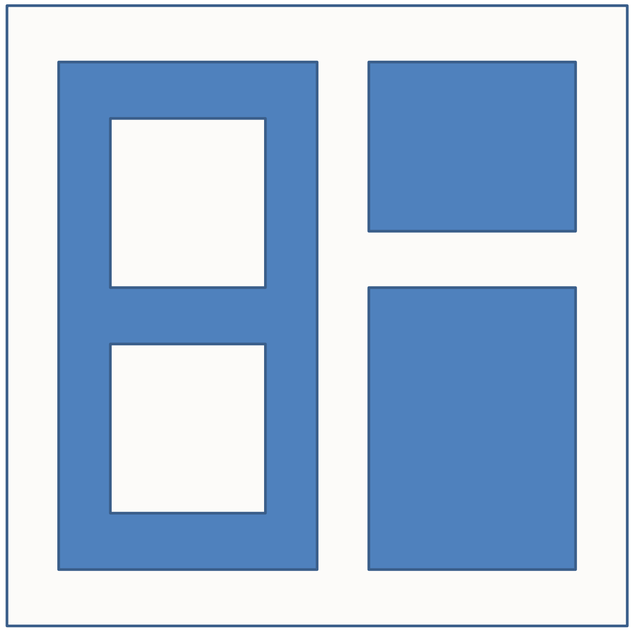


McGuffin 2012

Order Critical!

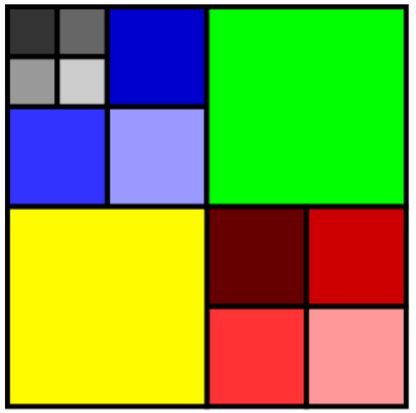


Implicit Layouts for Trees

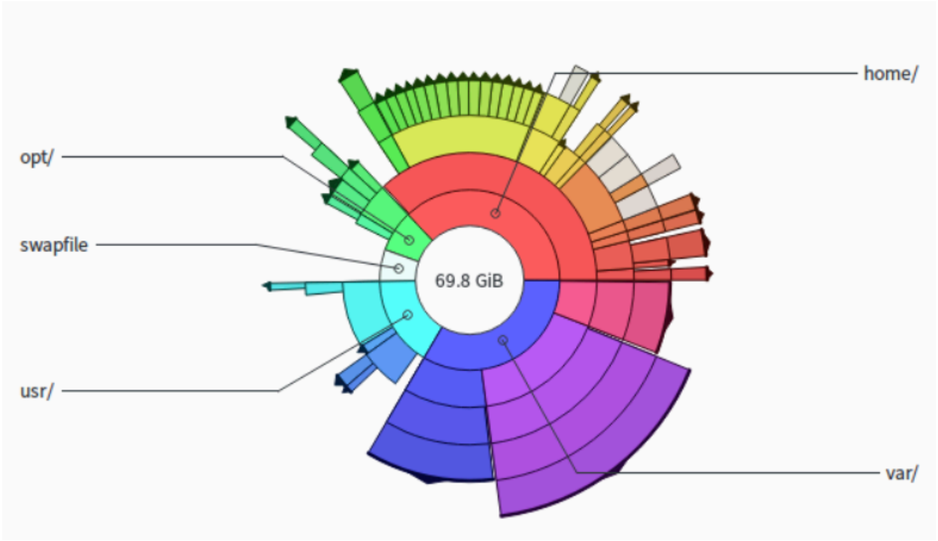


Implicit Layout Options

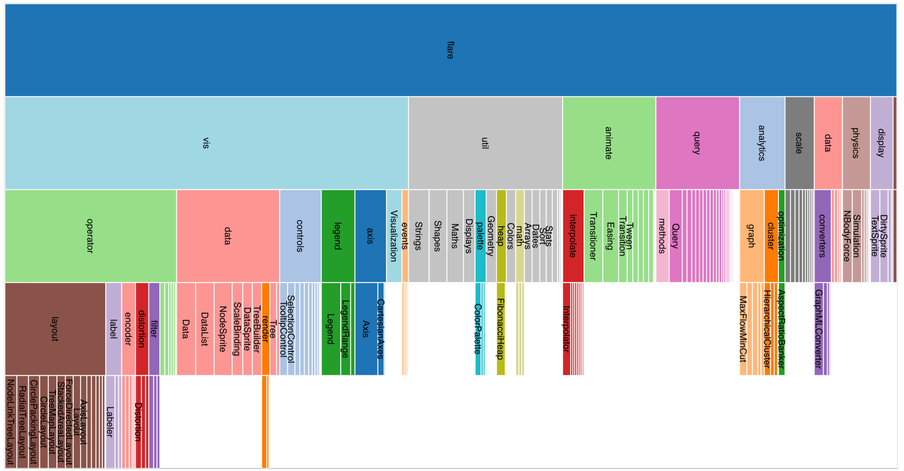
Treemap



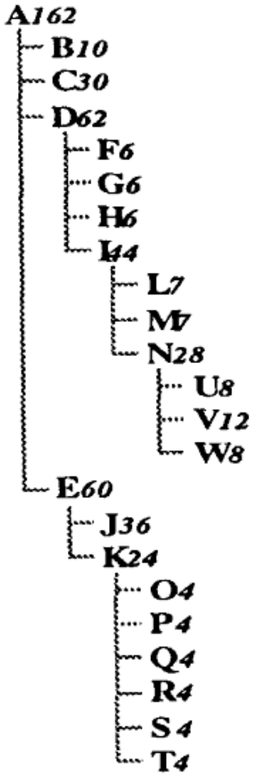
Sunburst

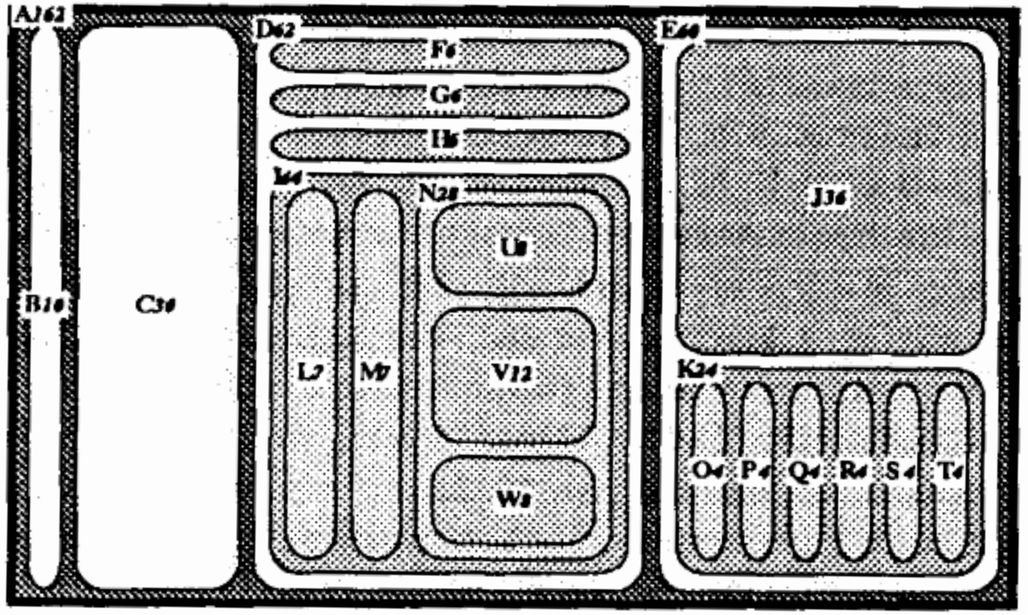


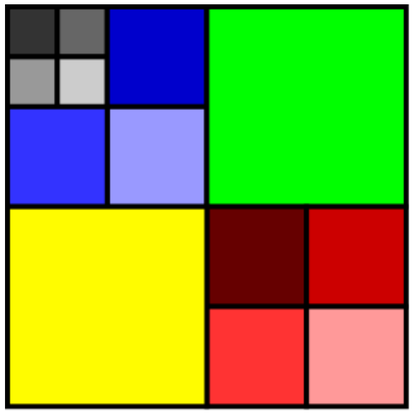
Icicle Plot

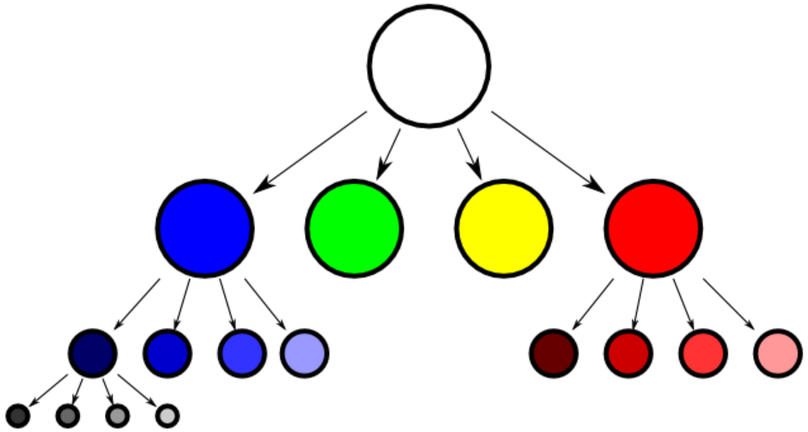


Tree Maps



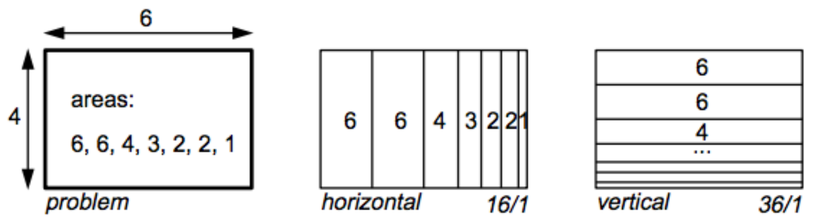






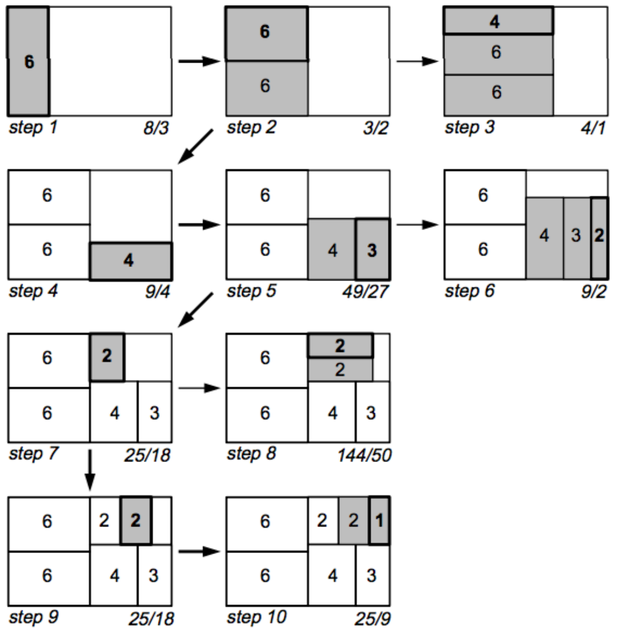
Johnson and Shneiderman 1991

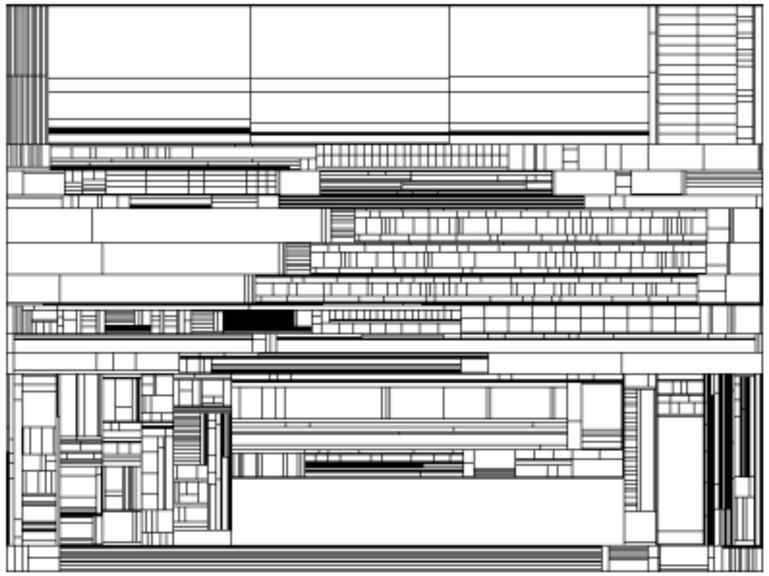
Squarified Treemaps



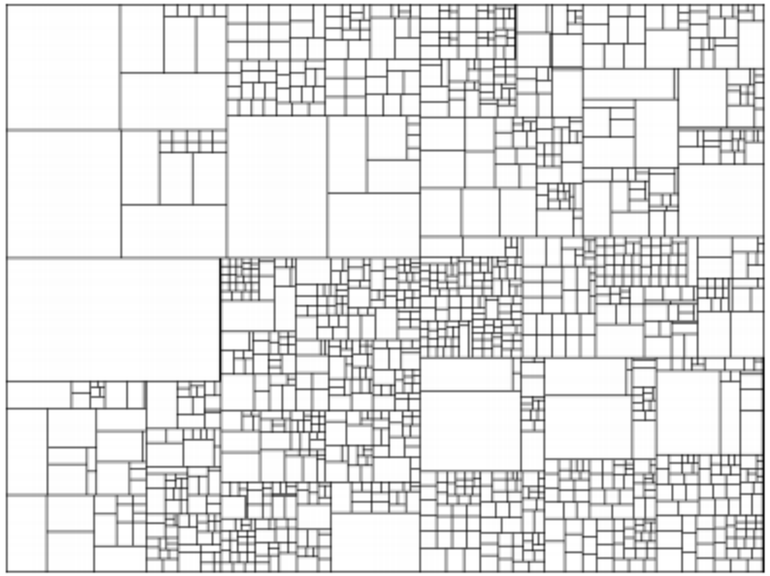
Original Algorithm lead to thin slices

Squarified treemaps [Bruls, Huizing, Van Wijk 2000]



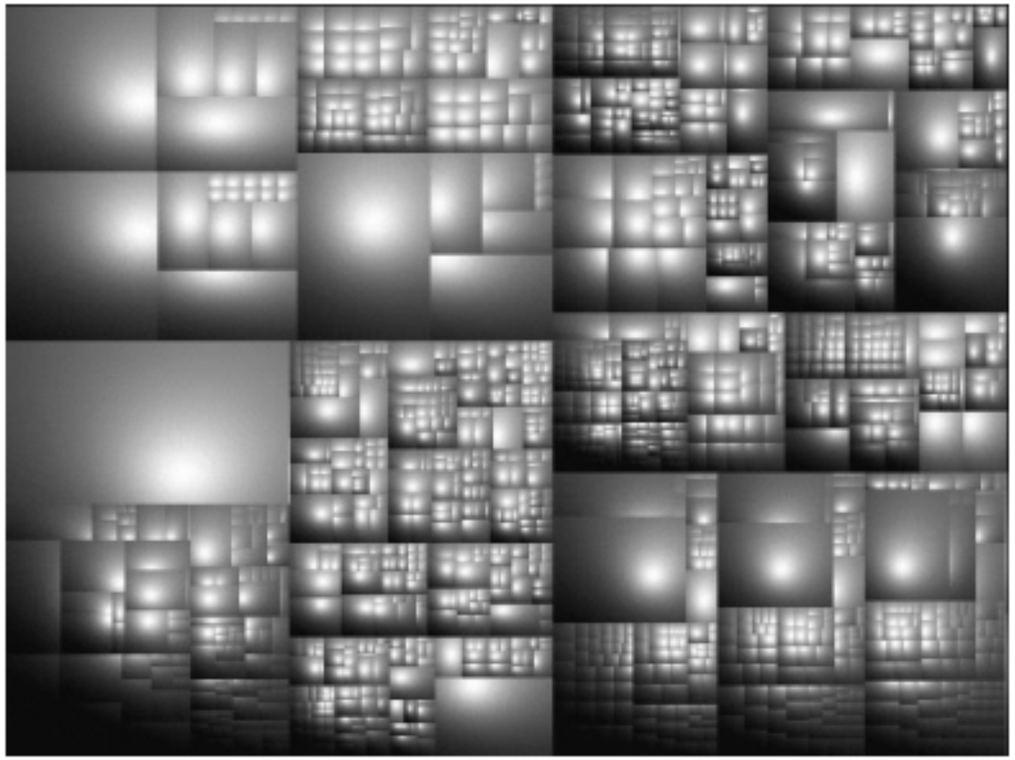


Before

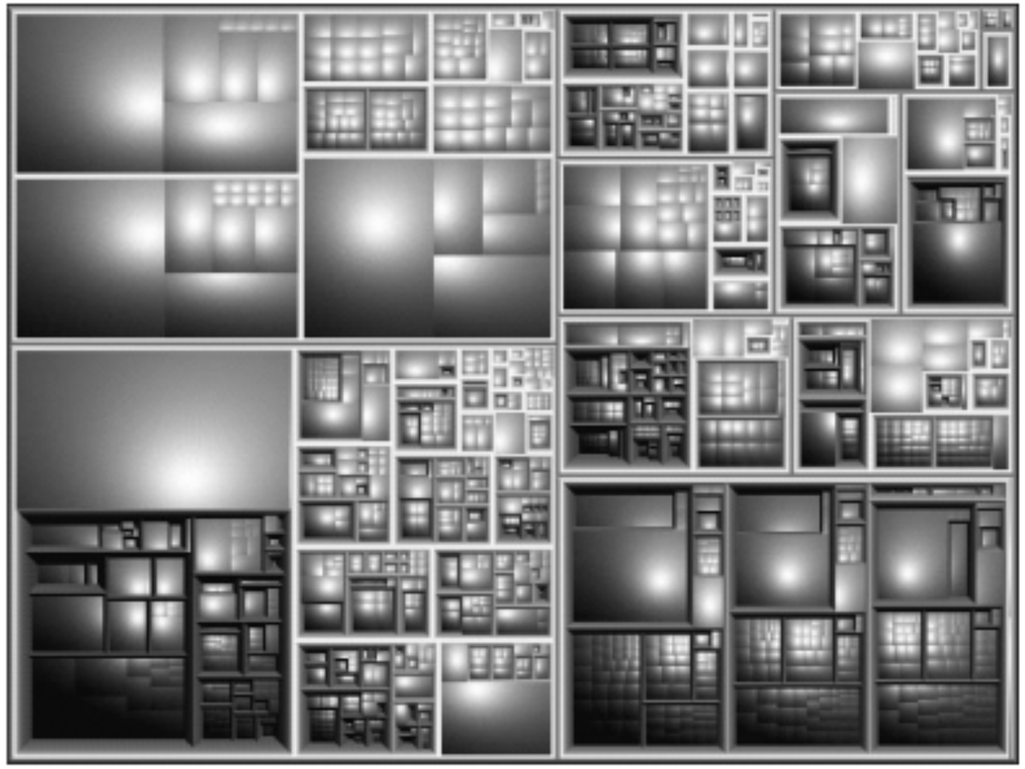


After

Seeing Tree Structure

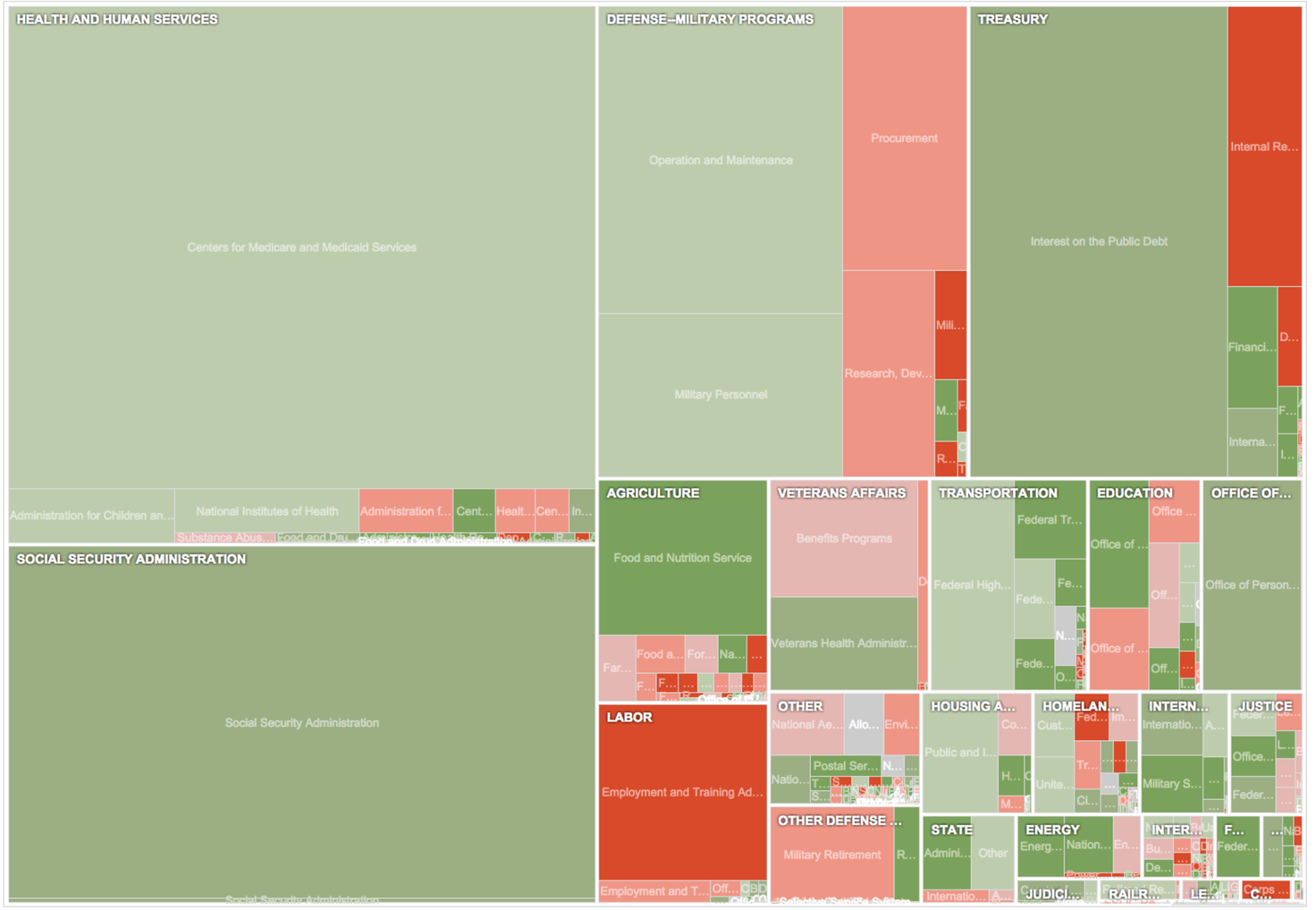


Unframed



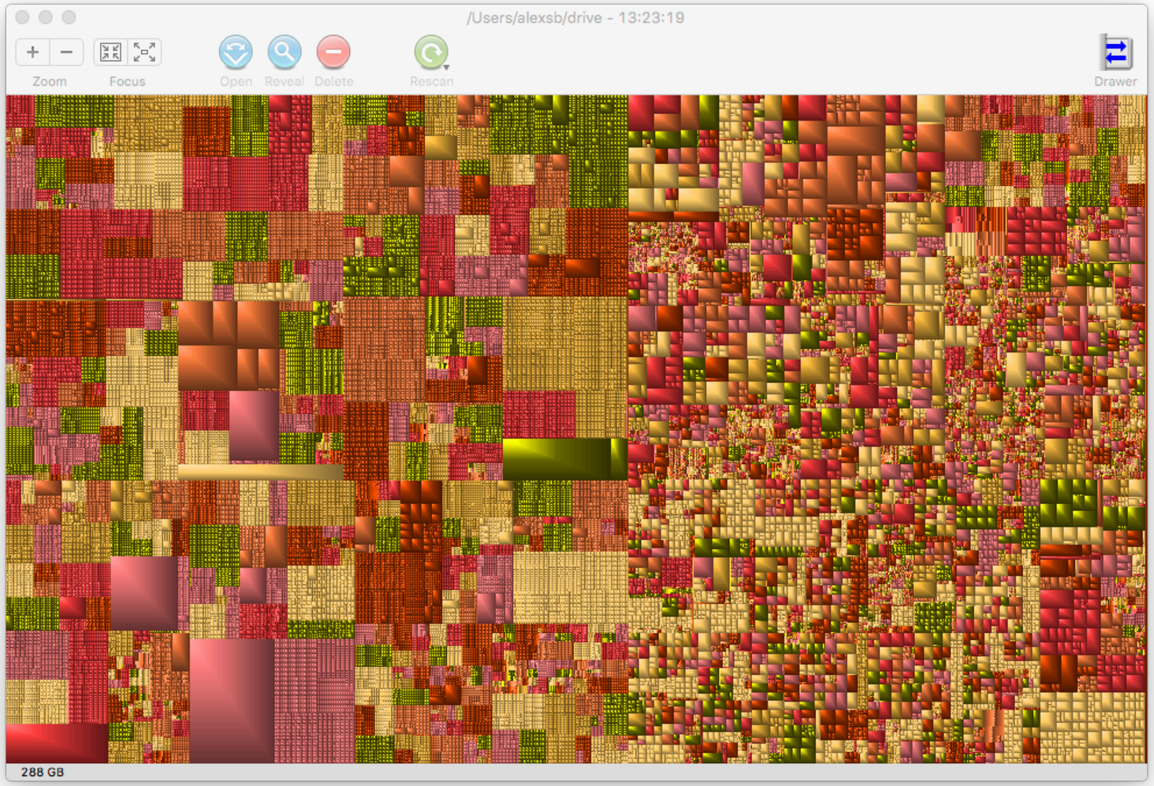
Framed

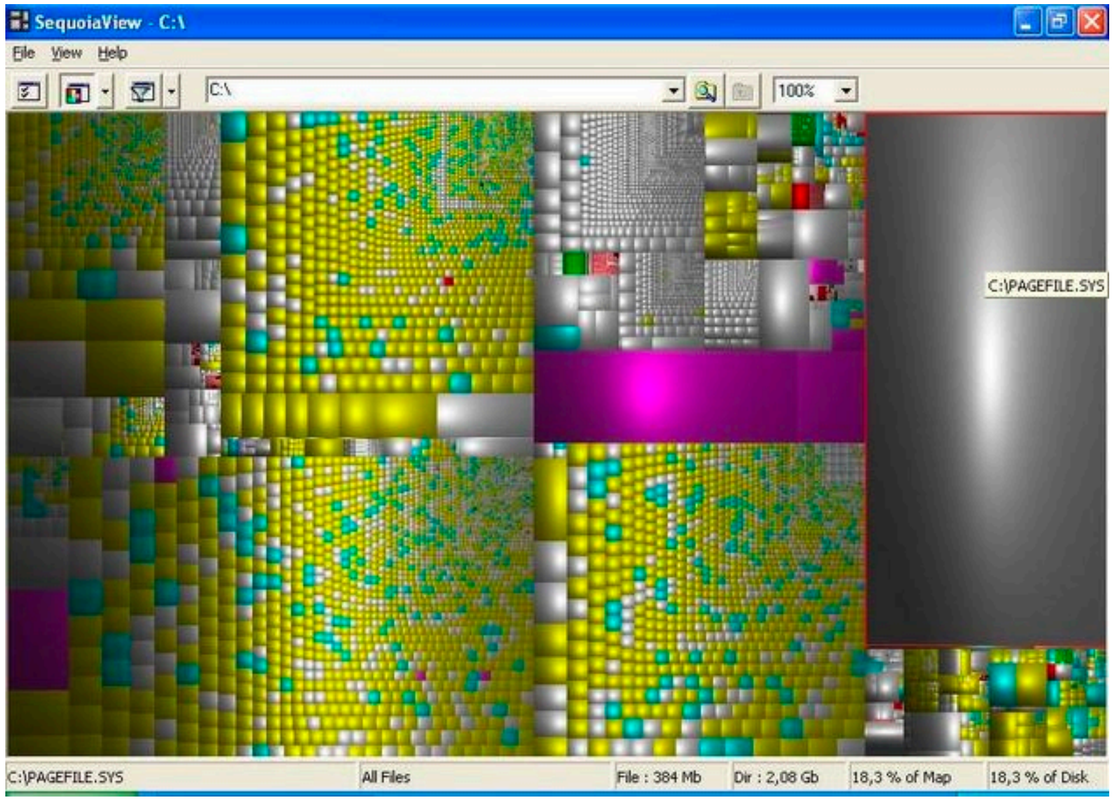
Zoomable Treemap



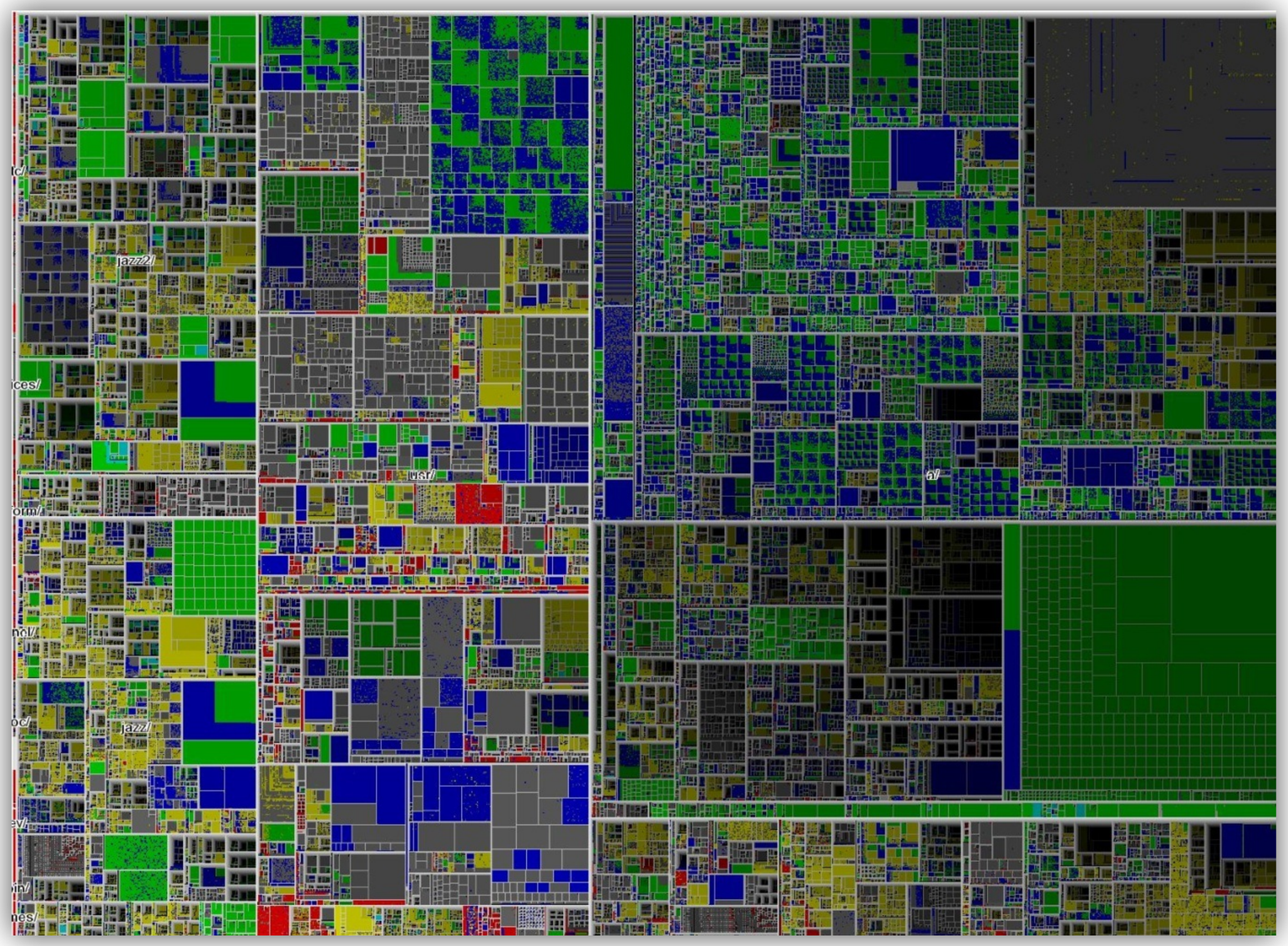
Software

Mac: GrandPerspective Windows: Sequoia View





Example: Interactive TreeMap of a Million Items

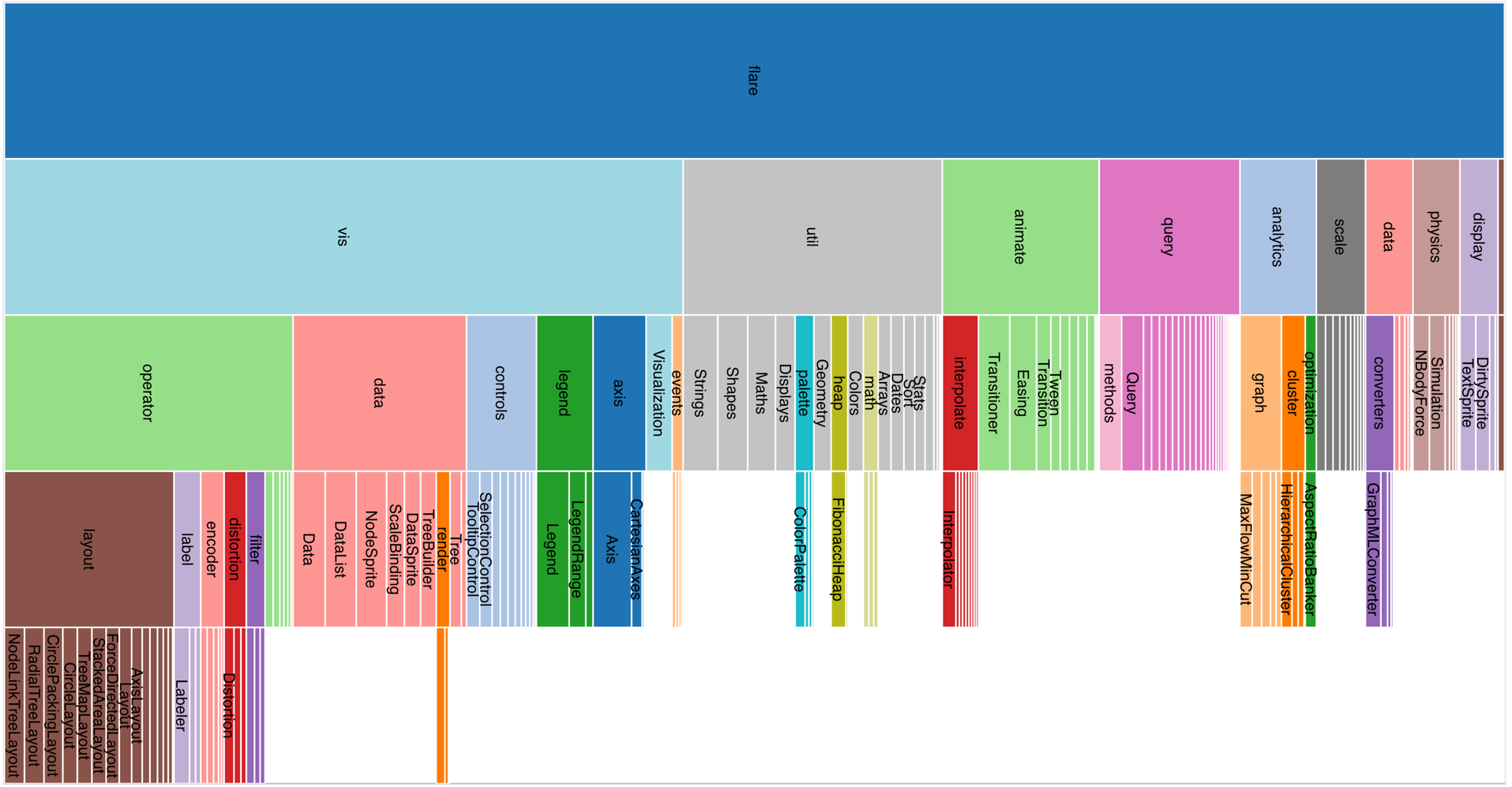


Fekete et al. 2002

Sunburst: Radial Layout

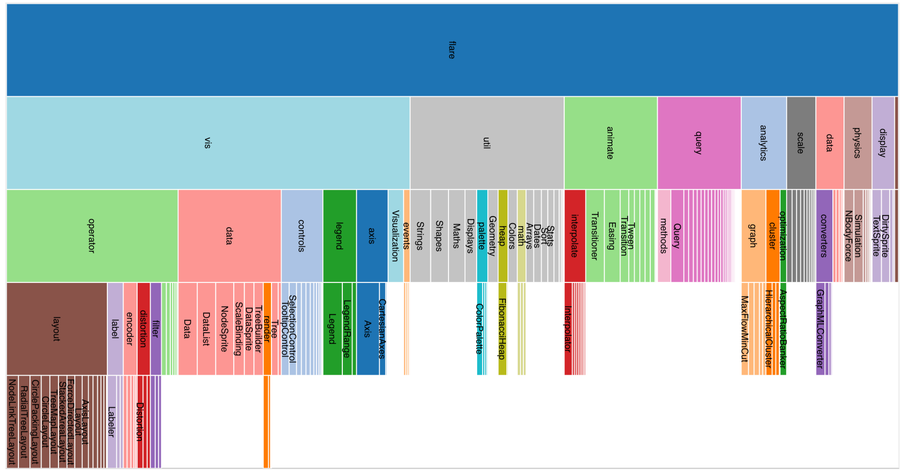
[Sunburst by John Stasko, Implementation in Caleydo by Christian Partl]

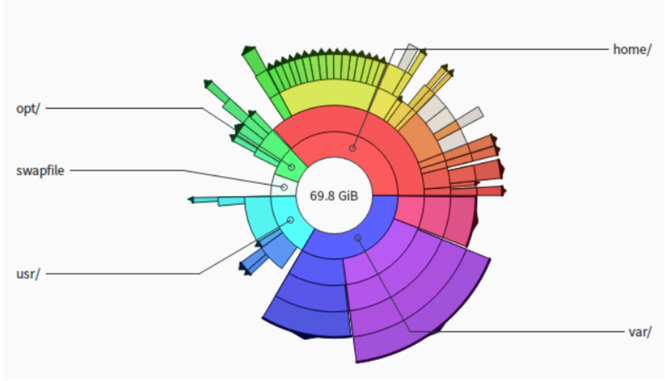
Icicle Plot

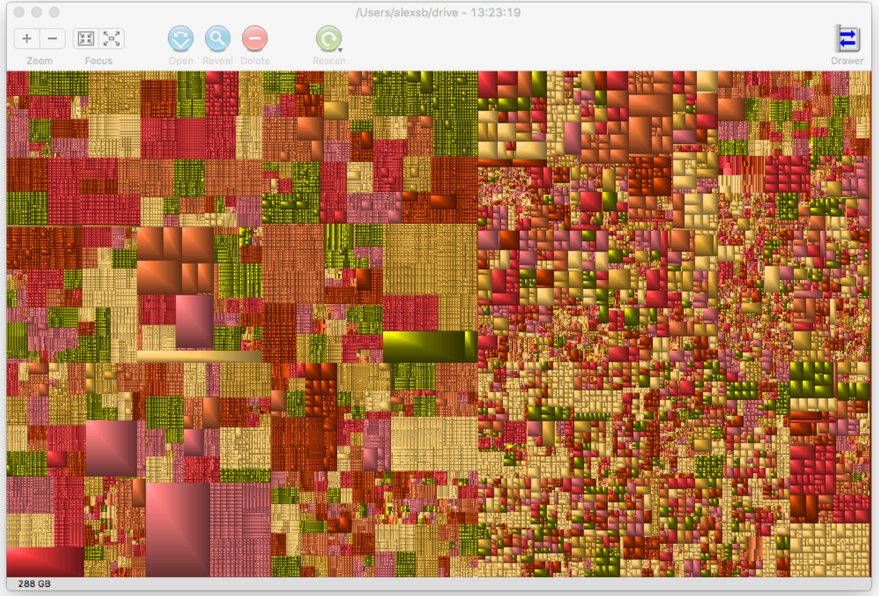


https://bl.ocks.org/mbostock/1005873

Differences? Pros, Cons?







Implicit Representations

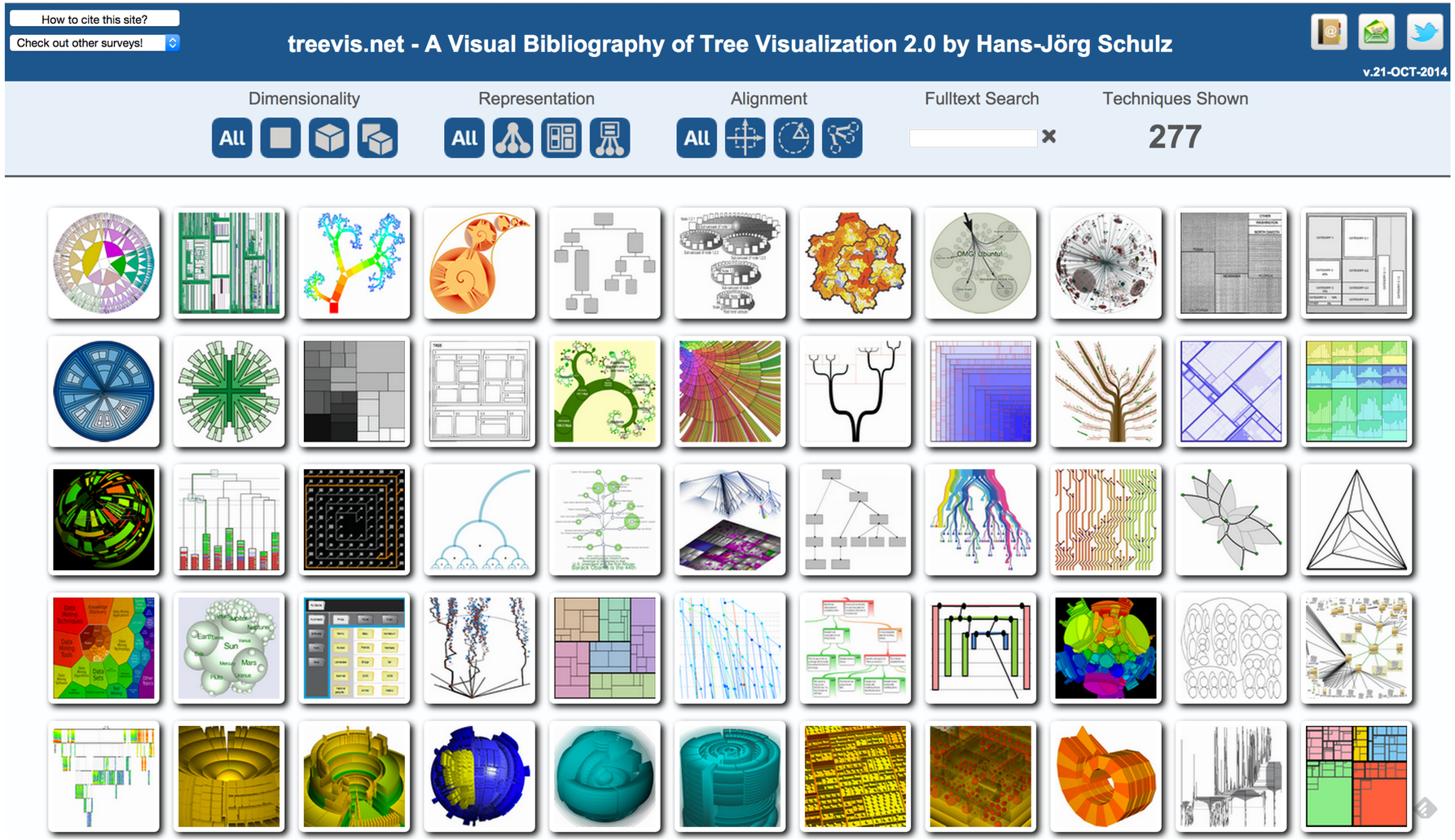
Pros:

- space-efficient because of the lack of explicitly drawn edges: scale well up to very large graphs
- in most cases well suited for ABTs on the node set
- depending on the spatial encoding also useful for TBTs

Cons:

- can only represent trees
- since the node positions are used to represent edges, they can no longer be freely arranged (e.g., to reflect geographical positions)
- useless to pursue any task on the edges

Tree Visualization Reference



Graph Tools & Applications

Gephi

http://gephi.org

Gephi
makes graphs handy

Download | Blog | Wiki | Forum | Support | Bug tracker

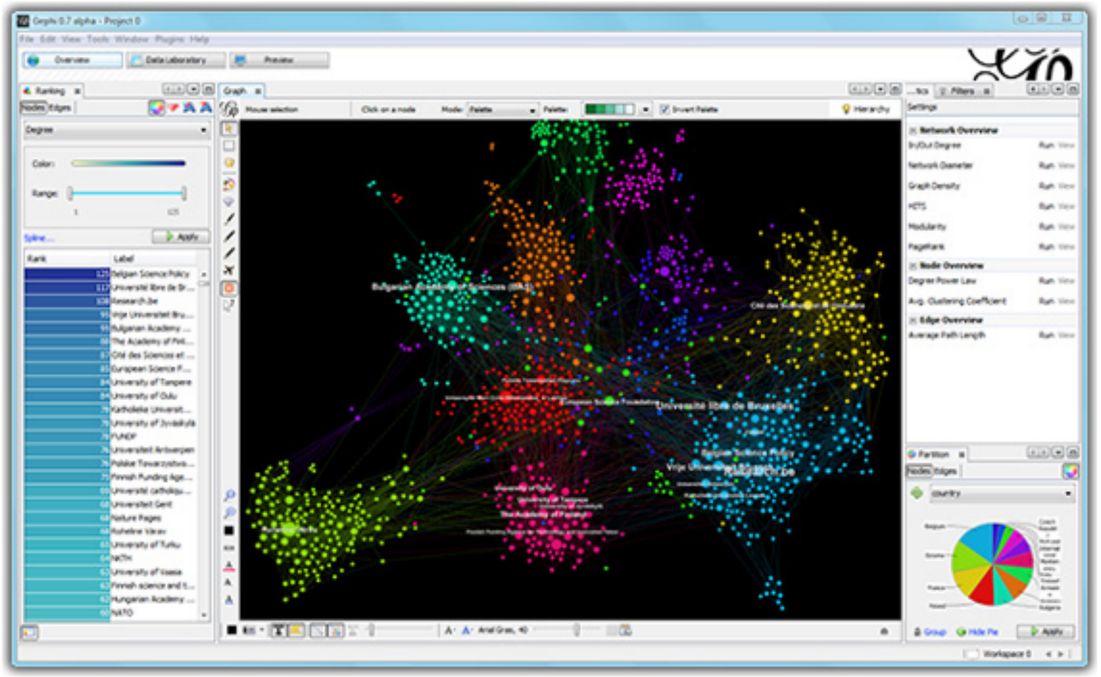
Home | Features | Plugins | Users | Developers | Partners

The Open Graph Viz Platform

Gephi is a visualization and exploration platform for all kinds of networks and complex systems, dynamic and hierarchical graphs.

Runs on Windows, Linux and Mac OS X. Gephi is open-source and free.

Learn More on Gephi Platform »



Download FREE
Gephi 0.7 alpha

Release Notes | System Requirements

- Features
- Quick start
- Screenshots
- Videos

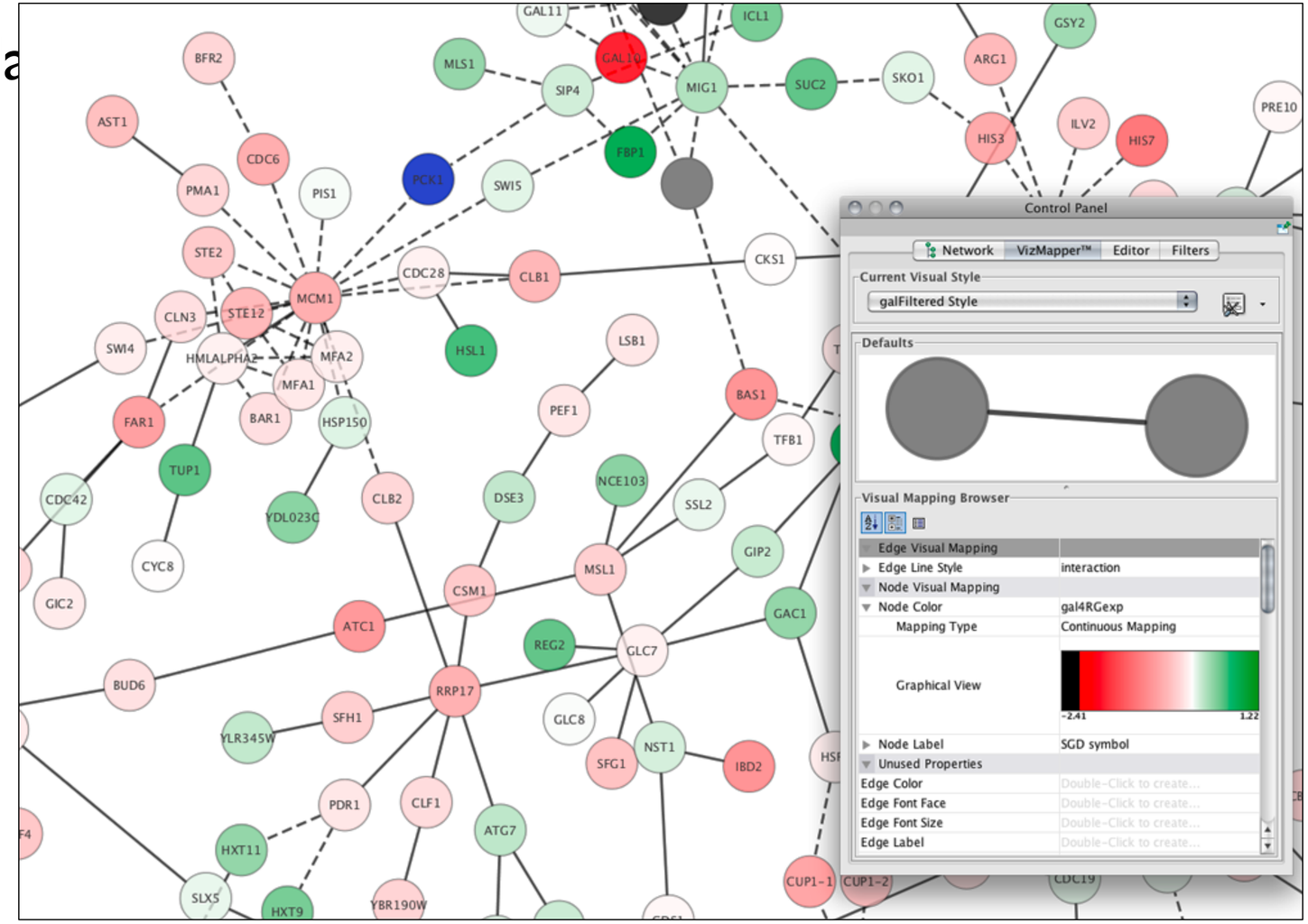
Gephi has been accepted again for Google Summer of Code! The program is the best way for students around the world to start contributing to an open-source project. Students, apply now for Gephi proposals. Come to the GSOC forum section and say Hi! to this topic.

Learn More »

Cytoscape

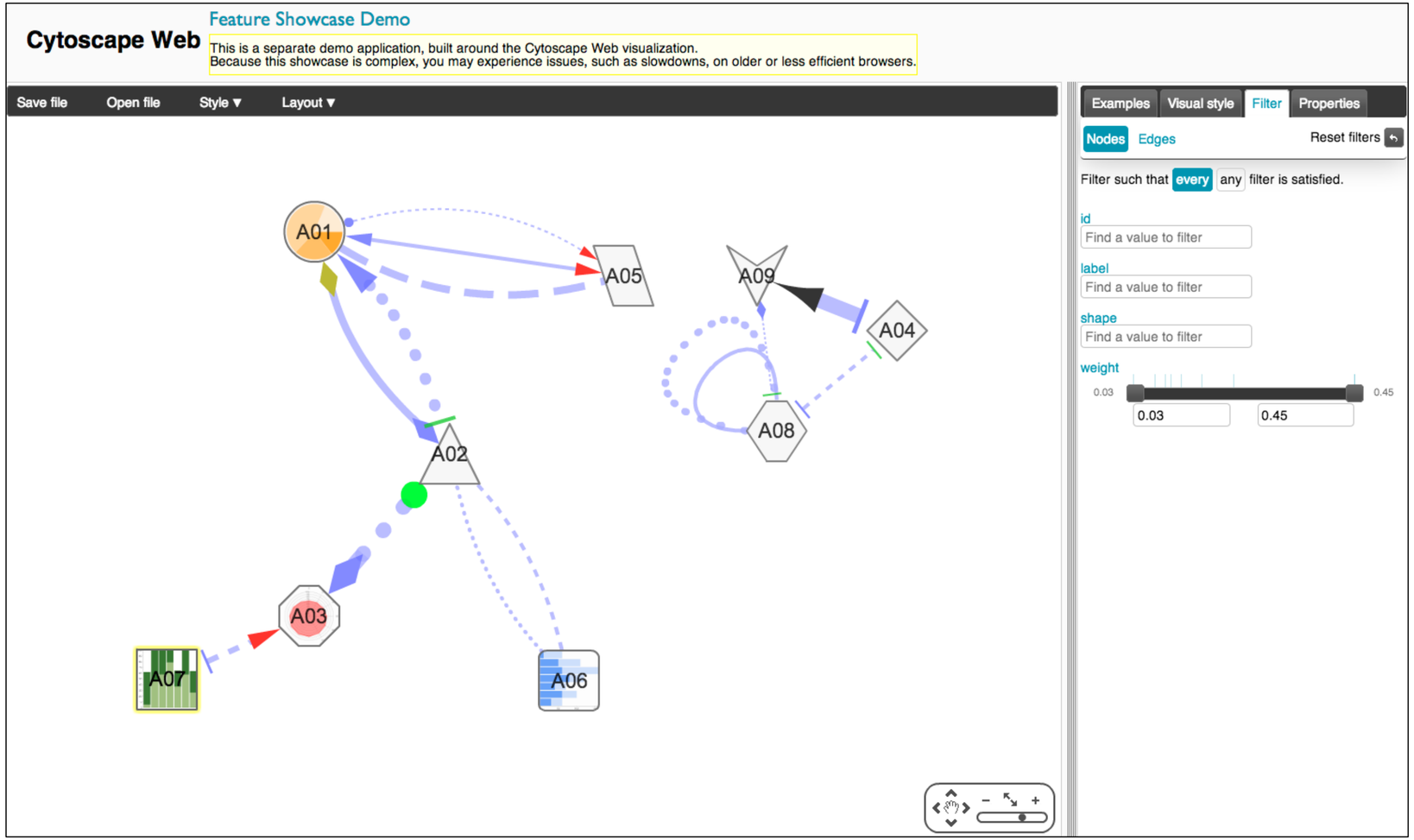
http://www.cytoscape.org/

Open source pla



Cytoscape Web

http://cytoscapeweb.cytoscape.org/



NetworkX

https://networkx.github.io/

NetworkX

NetworkX Home | Documentation | Download | Developer (Github)

High-productivity software for complex networks

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

Documentation
all documentation

Examples
using the library

Reference
all functions and methods

Features

- Python language data structures for graphs, digraphs, and multigraphs.
- Nodes can be "anything" (e.g. text, images, XML records)
- Edges can hold arbitrary data (e.g. weights, time-series)
- Generators for classic graphs, random graphs, and synthetic networks
- Standard graph algorithms
- Network structure and analysis measures
- Open source BSD license
- Well tested: more than 1800 unit tests, >90% code coverage
- Additional benefits from Python: fast prototyping, easy to teach, multi-platform

Versions

Latest Release

1.8.1 - 4 August 2013
downloads | docs | pdf

Development

1.9dev
github | docs | pdf
build passing
coverage 83%

Contact

Mailing list
Issue tracker
Developer guide

Graph Wrangling

https://origraph.github.io/