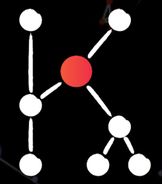


# GRAPH VISUALIZATION

KIRELL BENZI, PH.D

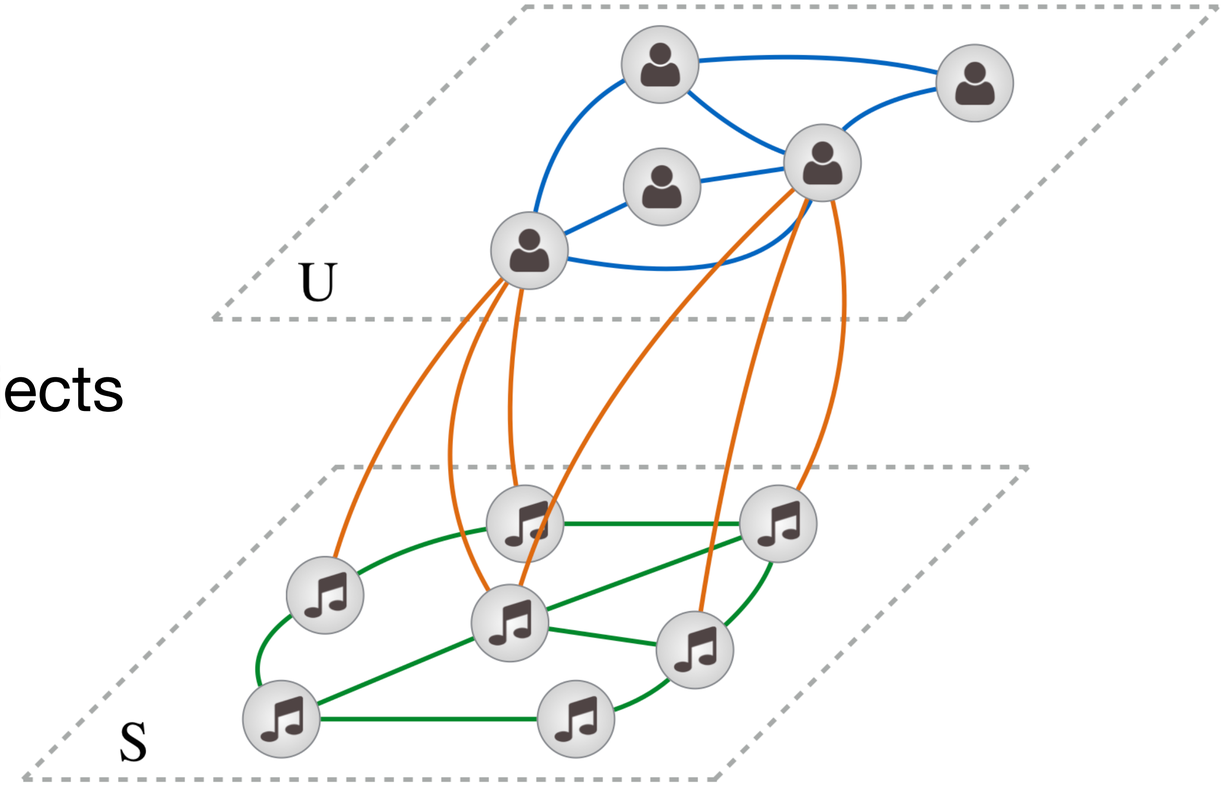
@KirellBenzi

Inspired by Lex

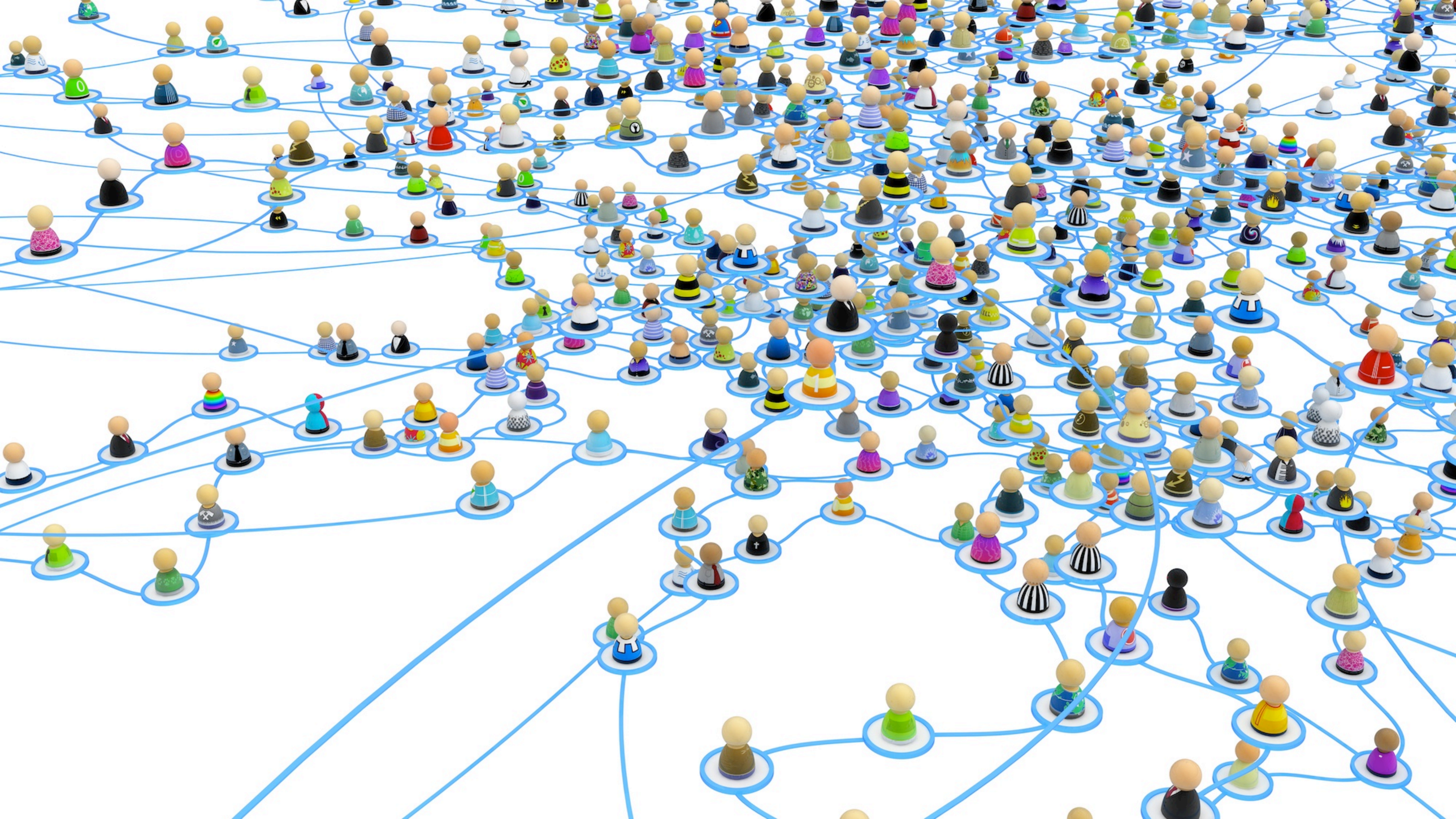
www.kirellbenzi.com

# What is a graph?

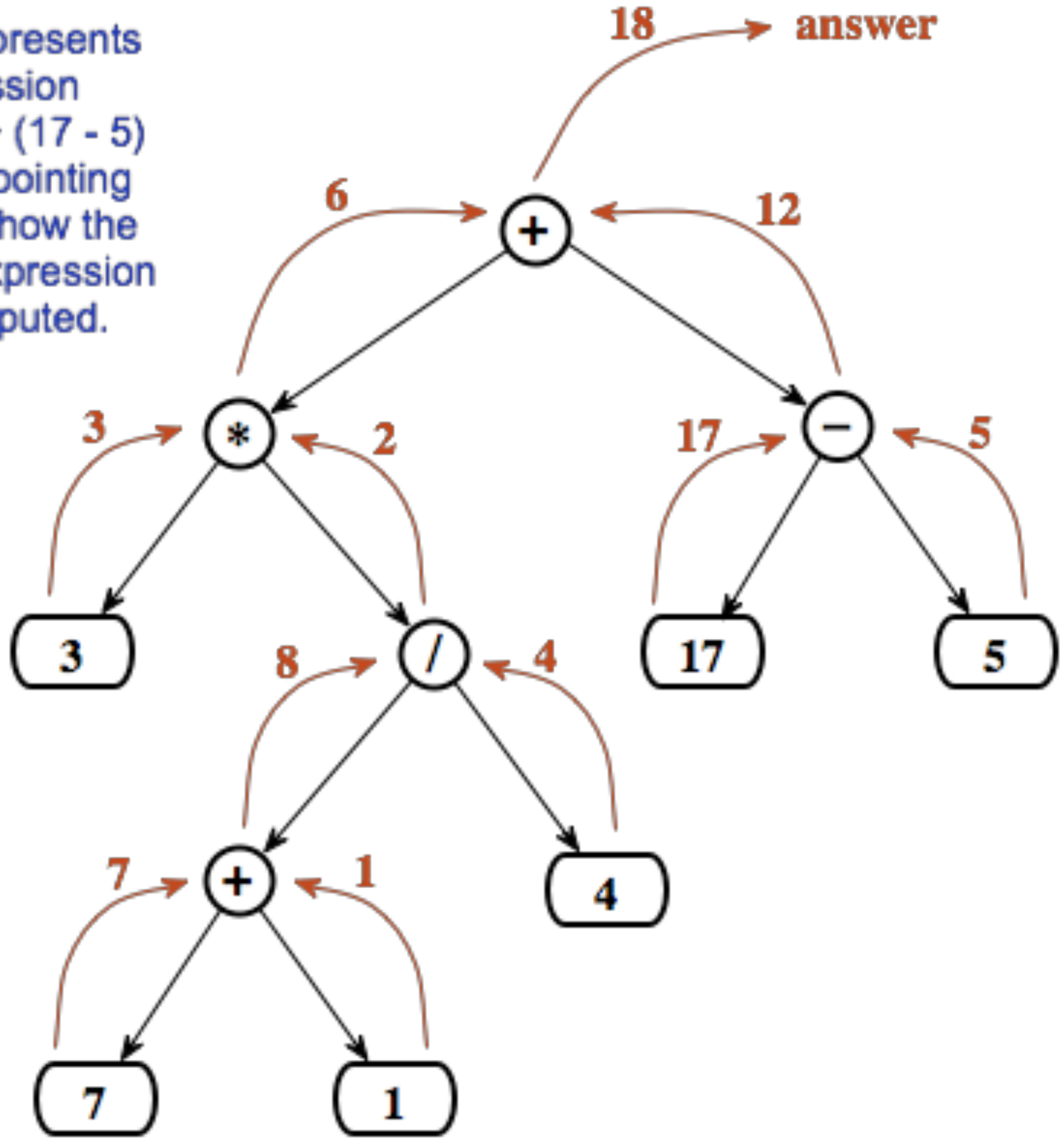
Mathematical tool to model
relationships between abstract objects



# Applications of graphs?



A tree that represents the expression
3 * ((7+1)/4) + (17 - 5)
The upward pointing arrows show how the value of the expression can be computed.



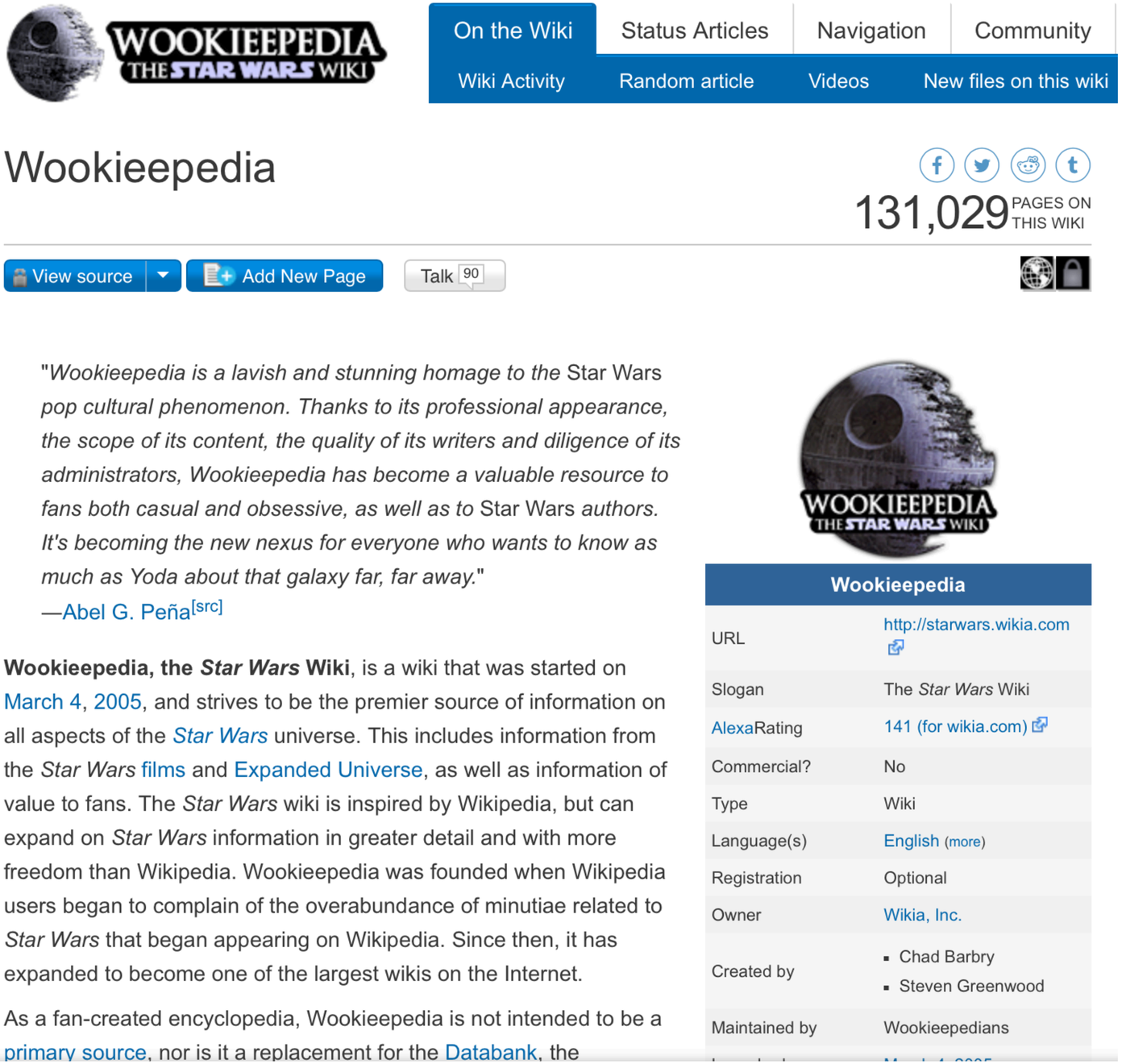


# STAR WARS

# WHY STAR WARS?

- Largest fictional universe ever created
- Free access to the data
- Leads to the creation of tools, algorithms and method to explore History with science

**Main reason:** I love Star Wars…



"*You were the Chosen One! It was said that you would destroy the Sith, not join them! Bring balance to the Force, not leave it in darkness! You were my brother, Anakin! I loved you!*"

—Obi-Wan Kenobi, to the fallen Anakin Skywalker[src]

**Anakin Skywalker** was a Force-sensitive Human male who served the Galactic Republic as a Jedi Knight and later served the Galactic Empire as the Sith Lord **Darth Vader**. Born to the slave Shmi Skywalker in 41.9 BBY, Anakin was conceived by midi-chlorians, the symbiotic organisms that allowed individuals to touch the Force, and he and his mother were brought to the desert planet of Tatooine to be the slaves of Gardulla the Hutt. They soon ended up as the property of the Toydarian Watto, and Skywalker exhibited exceptional piloting skills and a reputation for being able to build and repair anything even at a young age. In 32 BBY, Skywalker encountered the Jedi Qui-Gon Jinn and Padmé Amidala, and he helped them secure the parts they needed for their starship by winning the Boonta Eve Classic podracing event—only to learn that he had also won his freedom in doing so.

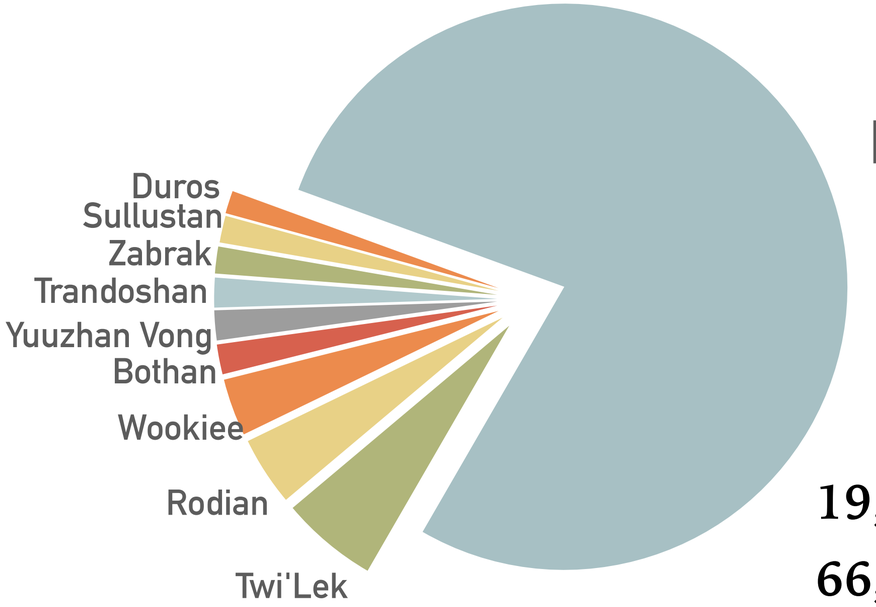
Reluctantly leaving his mother behind, Skywalker accompanied Jinn and his group to the Republic's capital of Coruscant and participated in the Battle of Naboo, helping to free Amidala's homeworld from the armies of the Trade Federation. While Jinn was killed during the battle, Jinn's student Obi-Wan Kenobi followed Jinn's wishes and took on Skywalker as his Padawan, and much of the Jedi Council believed that Skywalker was the Chosen One of Jedi prophecy: one who would bring balance to the Force by destroying the Sith Order. Skywalker and Kenobi had a number of adventures during the Padawan's decade of training to become a Jedi, but as the Separatist Crisis threatened to tear the Republic apart in 22 BBY, Skywalker was reunited with Amidala when he was assigned to protect her from assassins. The two grew close during the course of the assignment, though they were caught up in the Battle of Geonosis and the beginning of the Clone Wars between the Republic and the Confederacy of Independent Systems. Losing an arm during the fighting, Skywalker received a cybernetic replacement, and he and Amidala were secretly married in the days after the battle despite the Jedi Code's restrictions on romantic attachments.

During the Clone Wars, which raged for the next three years, Skywalker was granted the rank of Jedi Knight and became known to the public as the "Hero with No Fear." Taking on the Togruta Ahsoka Tano as his apprentice, Skywalker fought alongside Kenobi and his fellow Jedi in scores of battles, and his friendship with

| Anakin Skywalker | |
|---|---|
| **Biographical information** | |
| Homeworld | Tatooine[1] |
| Born | 41.9 BBY (7:4 BrS)[2] |
| Died | 4 ABY (39:3)[3] Death Star II over Endor[4] |
| **Physical description** | |
| Species | Human[1] |
| Gender | Male[1] |
| Height | 1.88 meters,[5] later 2.02 in armor[6] |
| Mass | 84 kilograms,[7] later 136 in armor[8] |
| Hair color | Blond[9] to brown,[5] la er none[4] |
| Eye color | Blue,[9] yellow (dark side)[10] |
| Skin color | Fair,[9] later pale[4] |
| Cybernetics | Cybernetic right arm,[11] later prosthetic arms and legs, and a life-support system[10] |
| **Chronological and political informatio** | |
| Era(s) | ▪ Rise of the Empire era[9]<br>▪ Rebellion era[12]<br>▪ New Republic era[4] |

# CREATING THE GRAPH OF CHARACTERS

- Fetch the hierarchy of categories under “Individuals”
- Fetch the biography of each character
- Detect in each bio other characters
- Create and store the network



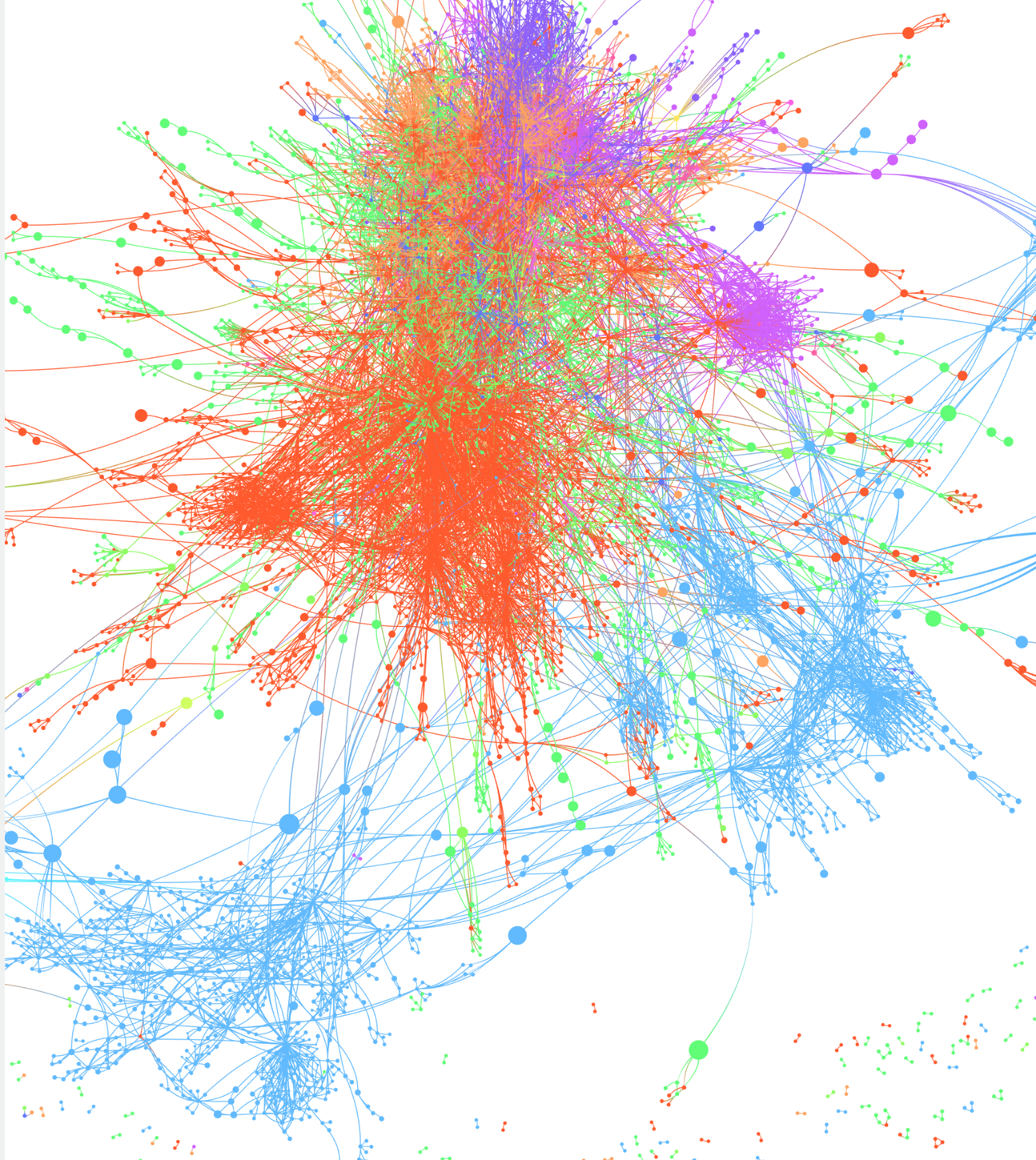

**19,613** characters

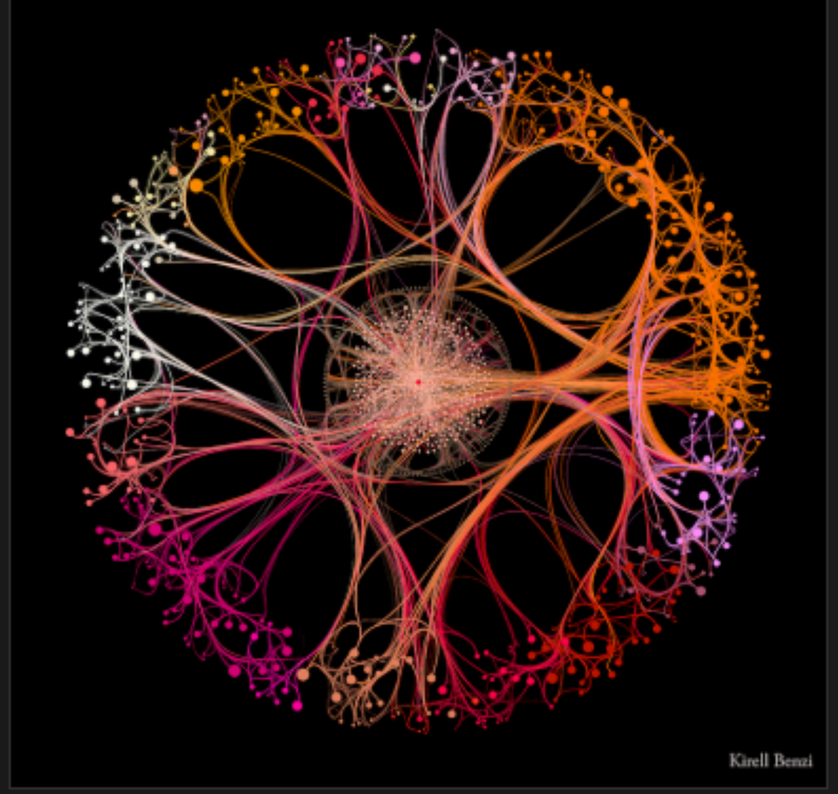
**66,425** links

# LABEL PROPAGATION

**Graph theory + machine learning**

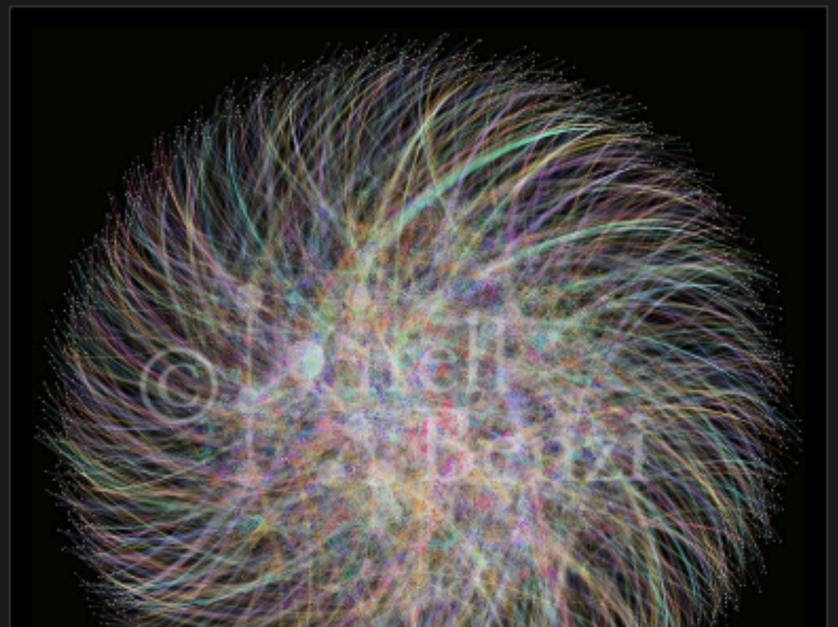
- Propagation of known information to undetermined neighbors (in black)
- Relies to the topology to infer missing values
- Can be also be used to improve opinion mining and sentiment analysis on Twitter

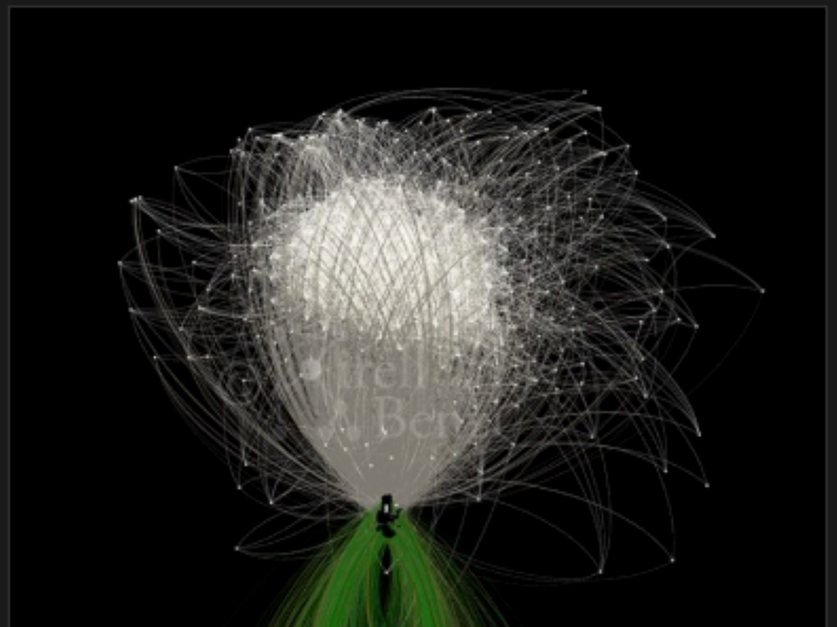



Kirell Benzi


© Kirell Benzi
Kirell Benzi


Kirell Benzi


© Kirell Benzi

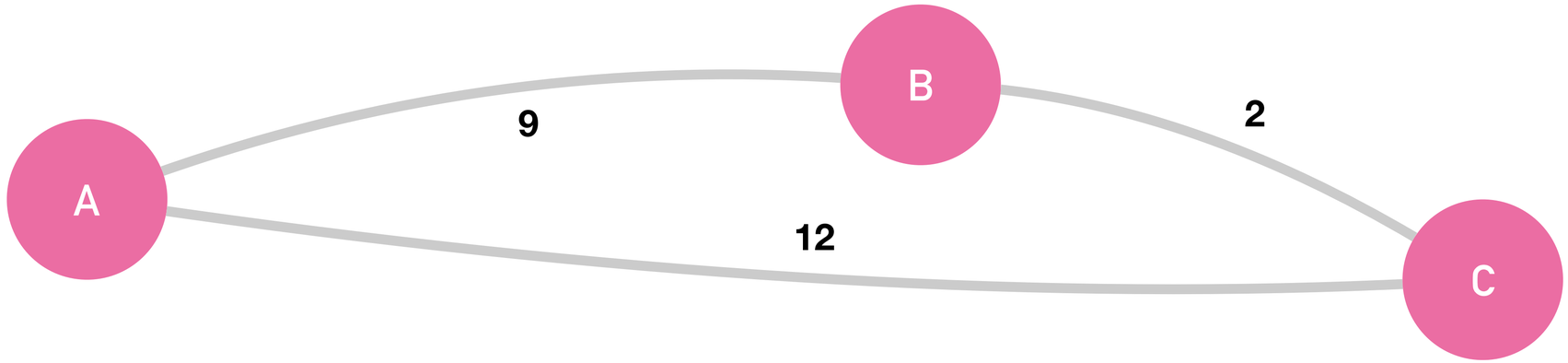


# Graph theory basics

A graph **G**(**V**,**E**,**W**) consists of a set of vertices V (also called nodes) and a set of edges E (also called links) connecting these vertices, and possibly a set of weights (W) associated with the edges.

Graph and Network are often used interchangeably

Edge weights: distance or strength?




**From Barabasi: http://barabasi.com/networksciencebook/chapter/2**

# Adjacency Matrix

N rows and N columns matrix representing the connectivity of the network

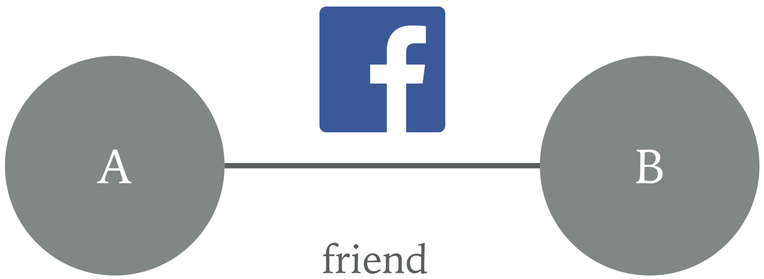
Aij = 1 if there is a link pointing from node j to node i

Aij = 0 if nodes i and j are not connected to each other

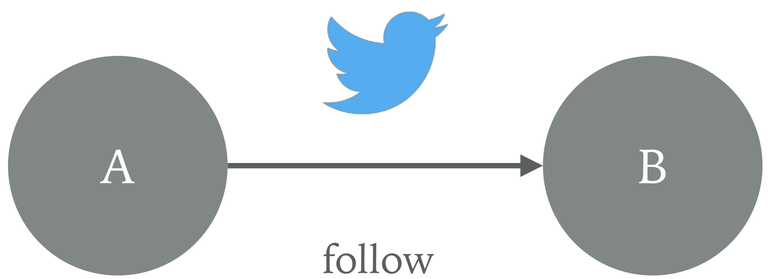
$$A_{ij} = \begin{matrix} A_{11} & A_{12} & A_{13} & A_{14} \\ A_{21} & A_{22} & A_{23} & A_{24} \\ A_{31} & A_{32} & A_{33} & A_{34} \\ A_{41} & A_{42} & A_{43} & A_{44} \end{matrix}$$

# Directed or not

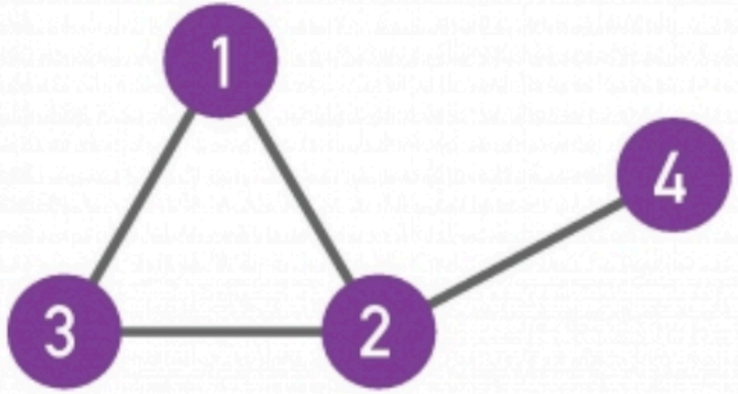
Undirected network




Directed network

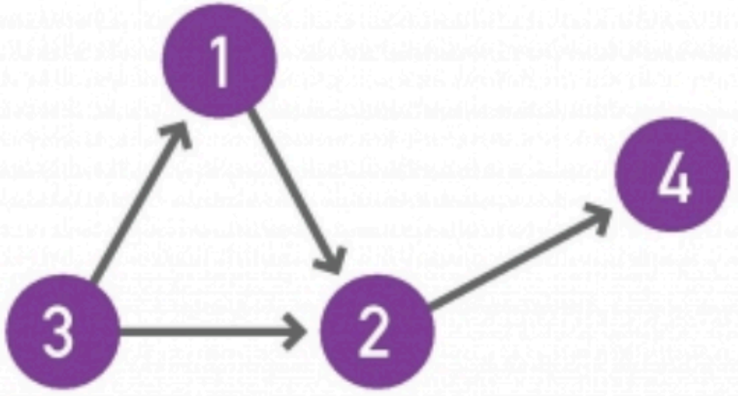


b. **Undirected network**




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

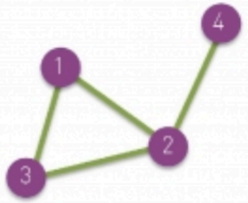
c. **Directed network**




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

# Some kinds of graphs

**a. Undirected**

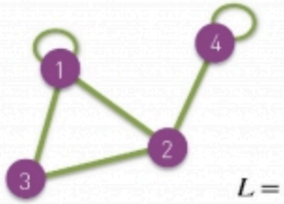


$$A_{ij} = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$$

$A_{ii} = 0 \qquad A_{ij} = A_{ji}$

$L = \frac{1}{2}\sum_{i,j=1}^{N} A_{ij} \qquad < k >= \frac{2L}{N}$

**b. Self-loops**

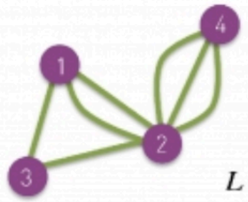


$$A_{ij} = \begin{pmatrix} 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix}$$

$\exists i,\ A_{ii} \neq 0 \qquad A_{ij} = A_{ji}$

$L = \frac{1}{2}\sum_{i,j=1,i\neq j}^{N} A_{ij} + \sum_{i=1}^{N} A_{ii} \qquad ?$

**c. Multigraph**
(undirected)

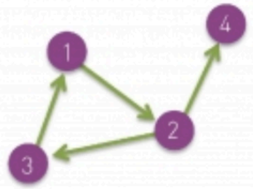


$$A_{ij} = \begin{pmatrix} 0 & 2 & 1 & 0 \\ 2 & 0 & 1 & 3 \\ 1 & 1 & 0 & 0 \\ 0 & 3 & 0 & 0 \end{pmatrix}$$

$A_{ii} = 0 \qquad A_{ij} = A_{ji}$

$L = \frac{1}{2}\sum_{i,j=1}^{N} A_{ij} \qquad < k >= \frac{2L}{N}$

**d. Directed**

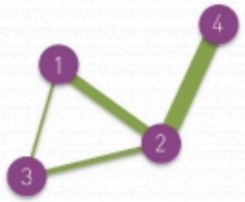


$$A_{ij} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

$A_{ij} \neq A_{ji}$

$L = \sum_{i,j=1}^{N} A_{ij} \qquad < k >= \frac{L}{N}$

**e. Weighted**
(undirected)



$$A_{ij} = \begin{pmatrix} 0 & 2 & 0.5 & 0 \\ 2 & 0 & 1 & 4 \\ 0.5 & 1 & 0 & 0 \\ 0 & 4 & 0 & 0 \end{pmatrix}$$

$A_{ii} = 0 \qquad A_{ij} = A_{ji}$

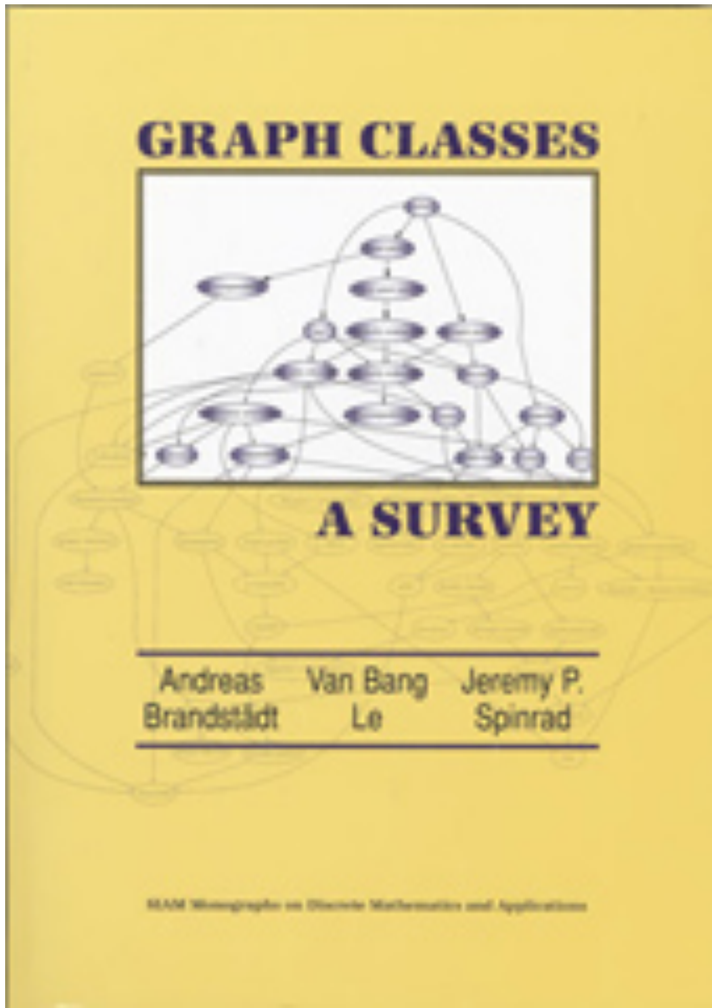
$< k >= \frac{2L}{N}$

**f. Complete Graph**
(undirected)

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

$A_{ii} = 0 \qquad A_{i\neq j} = 1$

$L = L_{\max} = \frac{N(N-1)}{2} \qquad < k >= N - 1$



# Knowledge graph?

## How to build a knowledge graph?

5

12

I prototyped a tiny search engine with pagerank that worked on my computer. I am interested in building a knowledge graph on top of it, and it should return only queried webpages that are within the right context, similarly to how Google found relevant answers to search questions. I saw a lot of publicity around knowledge graph but not a lot of literature and almost no pseudocode like guideline of building one. Does anyone know good references on how such knowledge graph works internally, so there will be no need to create models about a knowledge graph?

algorithm search graph artificial-intelligence

share edit close flag

asked Apr 5 '15 at 19:02

Pippi

850 ●2 ●20 ●37

add a comment

start a bounty

## 1 Answer

active oldest votes

20

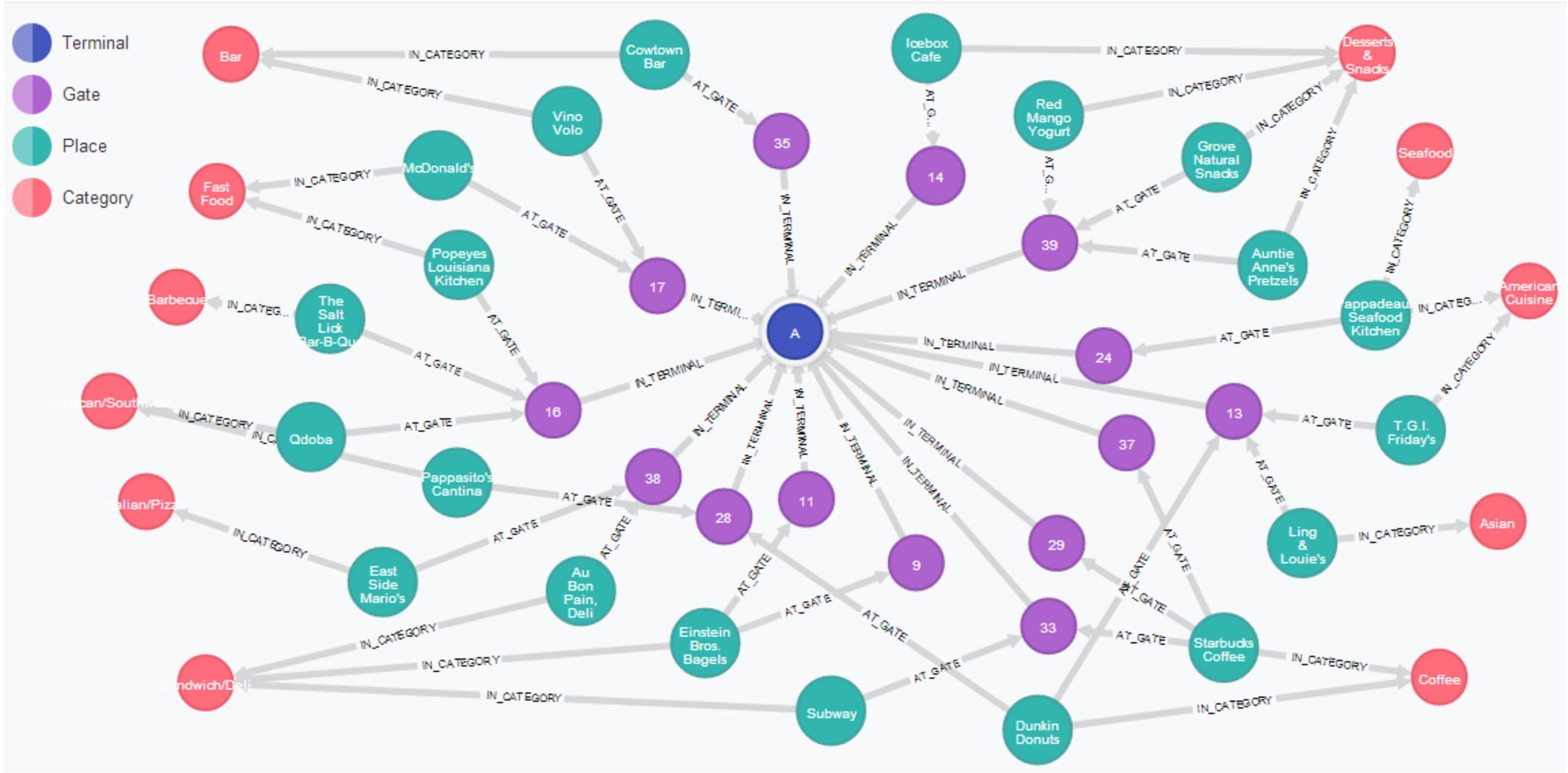
Knowledge graph is a buzzword. It is a sum of models and technologies put together to achieve a result. The first stop on your journey starts with Natural language processing, Ontologies and Text mining. It is a wide field of artificial intelligence, go here for a research survey on the field.

Before building your own models, I suggest you try different standard algorithms using dedicated toolboxes such as gensim. You will learn about tf-idf, LDA, document feature vectors, etc.

I am assuming you want to work with text data, if you want to do image search using other images it is different. Same for the audio part.

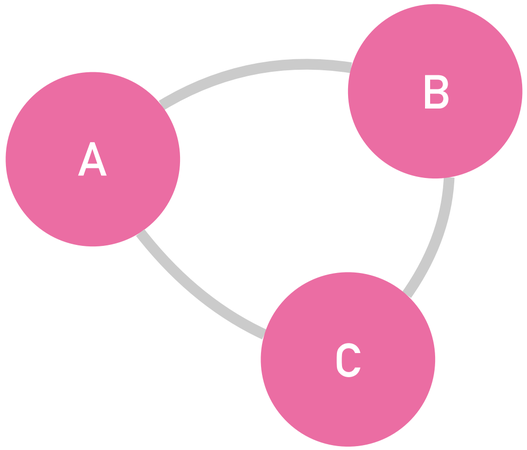
# Graph databases

```
CYPHER MATCH p = (:Category)<--(:Place)-[*]->(:Terminal {name:'A'}) RETURN p
```

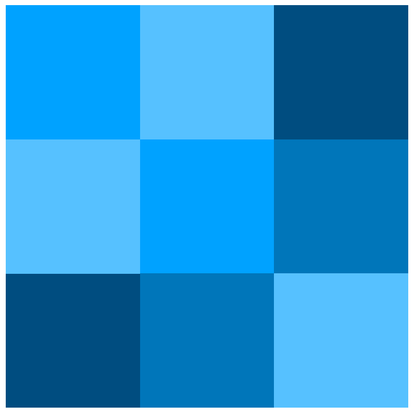


# Graph visualizations

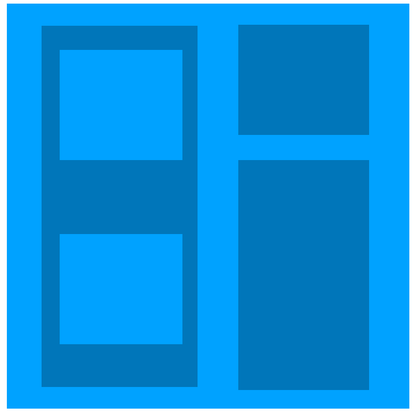
# Graph visual encodings




**Node-link diagram**



**Matrix form**



**Implicit form**

# Choosing the right representation?

Depends on the task, **attribute-based (ABT)** and/or **topology-based (TBT)**

**Localize** – find a single or multiple nodes/edges

- ABT: Find edges of a certain type
- TBT: Get the neighborhood of a given node

**Quantify** – count or estimate a numerical property of the graph

- ABT: Get number of edges
- TBT: Get the number of outboud links from a node

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

- ABT: Sort all edges by distance (or weight)
- TBT: Get the shortest paths from a given node

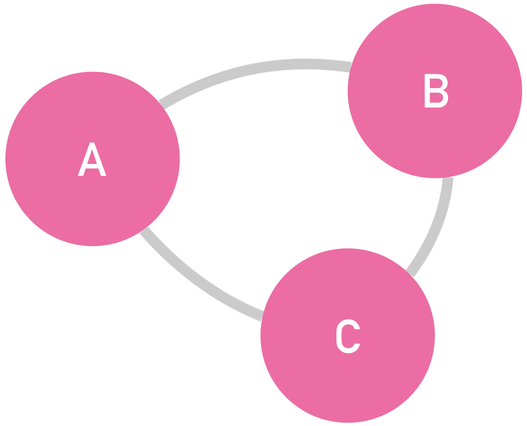
**[Lex]**

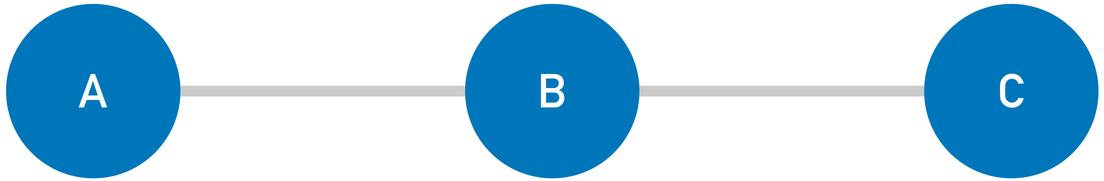
# Node-link diagram

Explicit graph representation

Vertex = point

Edge = line or arc



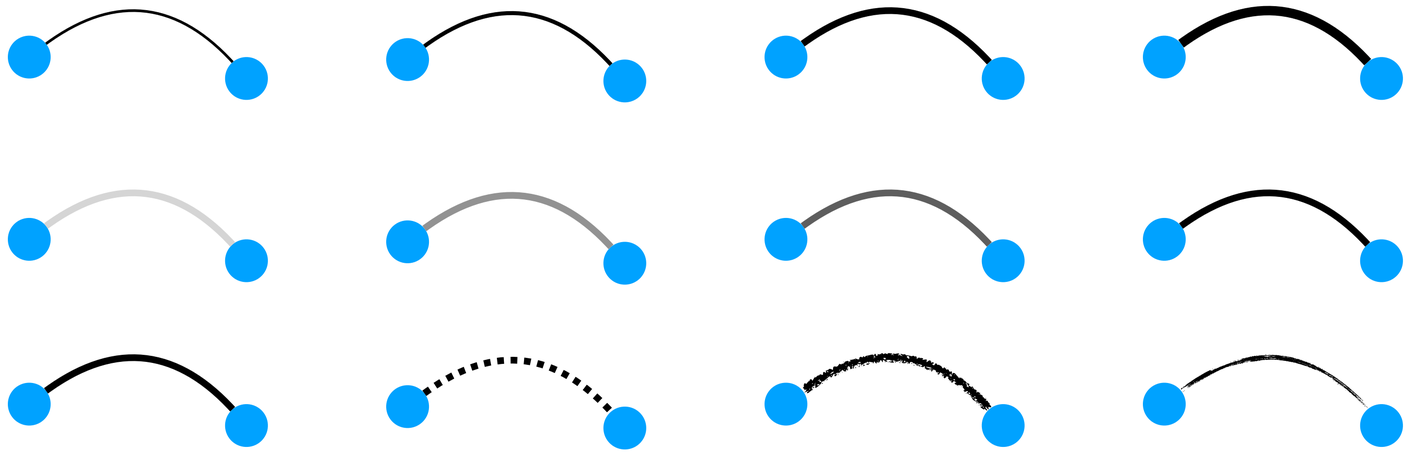



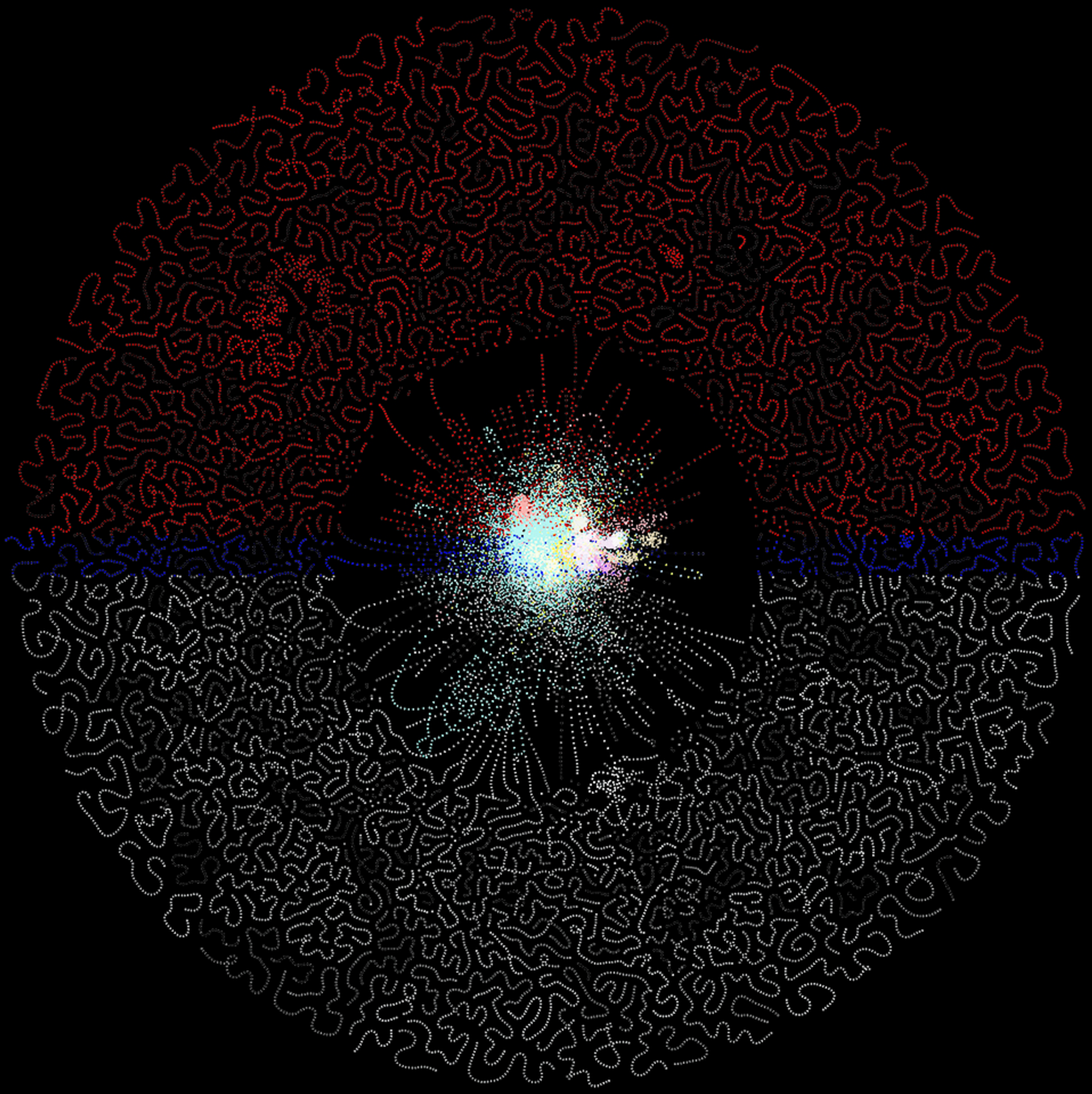
# Edge attribute encoding

**Width**: quantitative

**Saturation**: ordinal

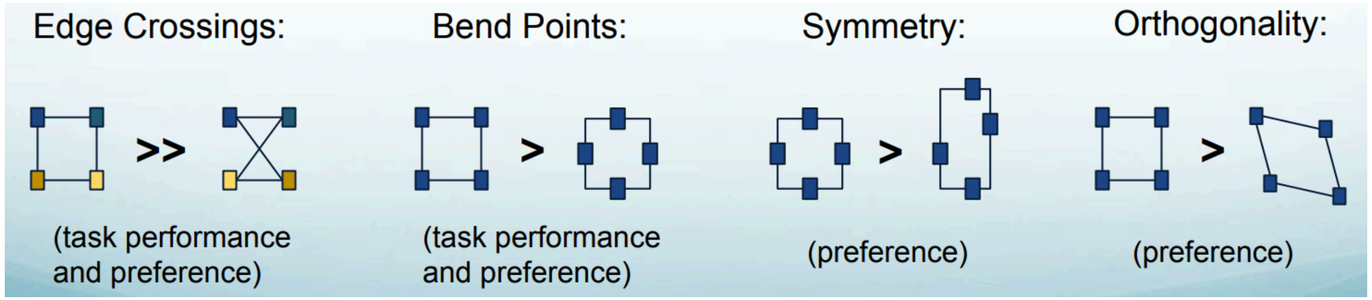
**Style**: nominal





Kirell Benzi

# Criteria for good layout




**[Hindalong]**

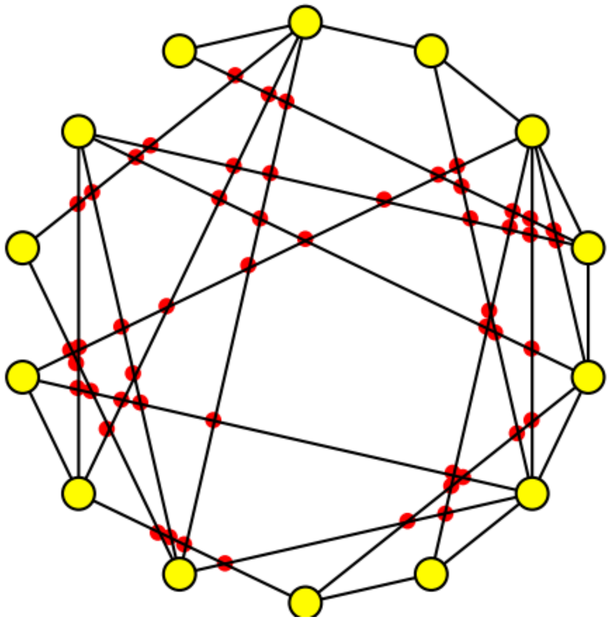
Minimized distance of neighboring nodes

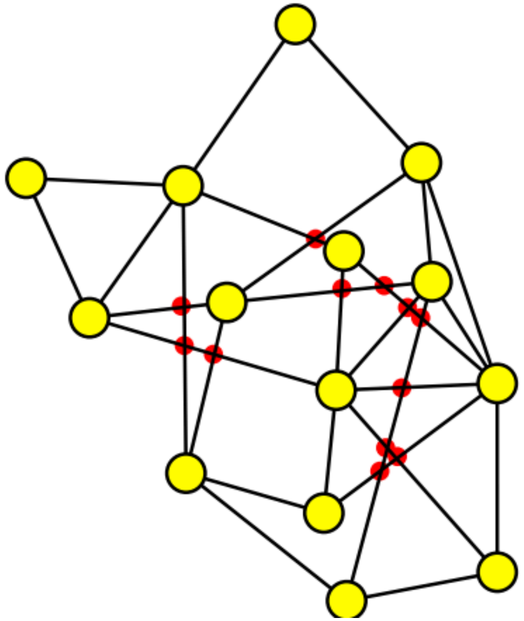
Uniform edge length

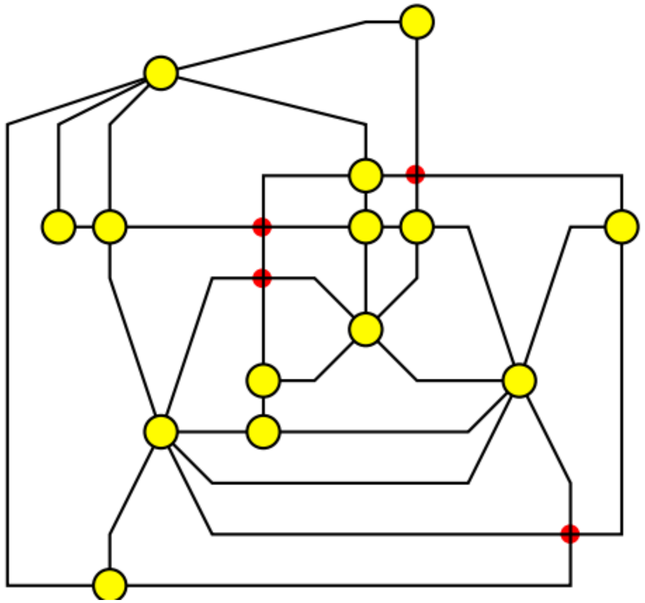
Aspect ratio about 1 (not too long and not too wide)

Minimized drawing area

# Conflicting criteria



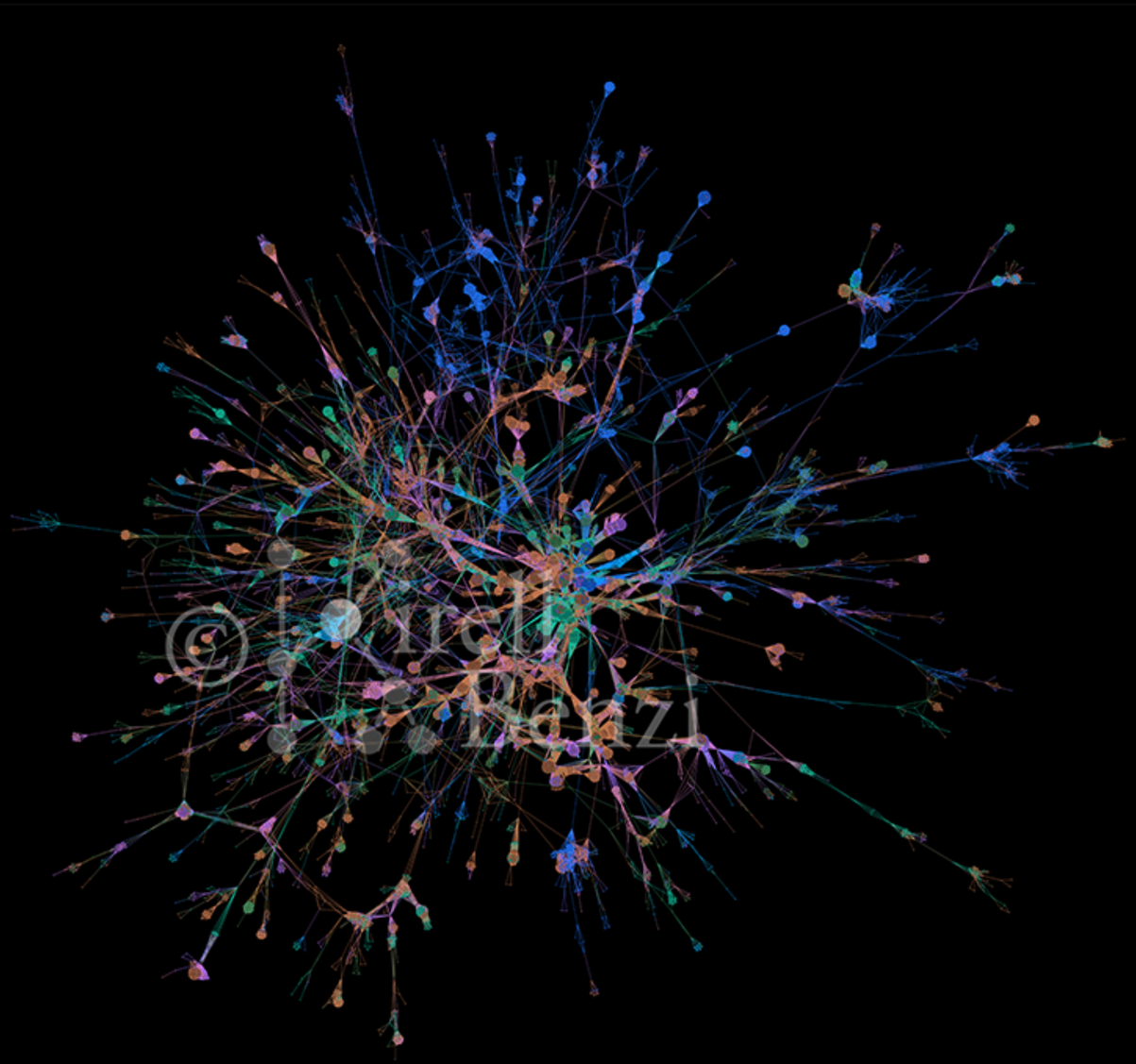




[Buchheim et al.]

# Design critique


Kirell Benzi



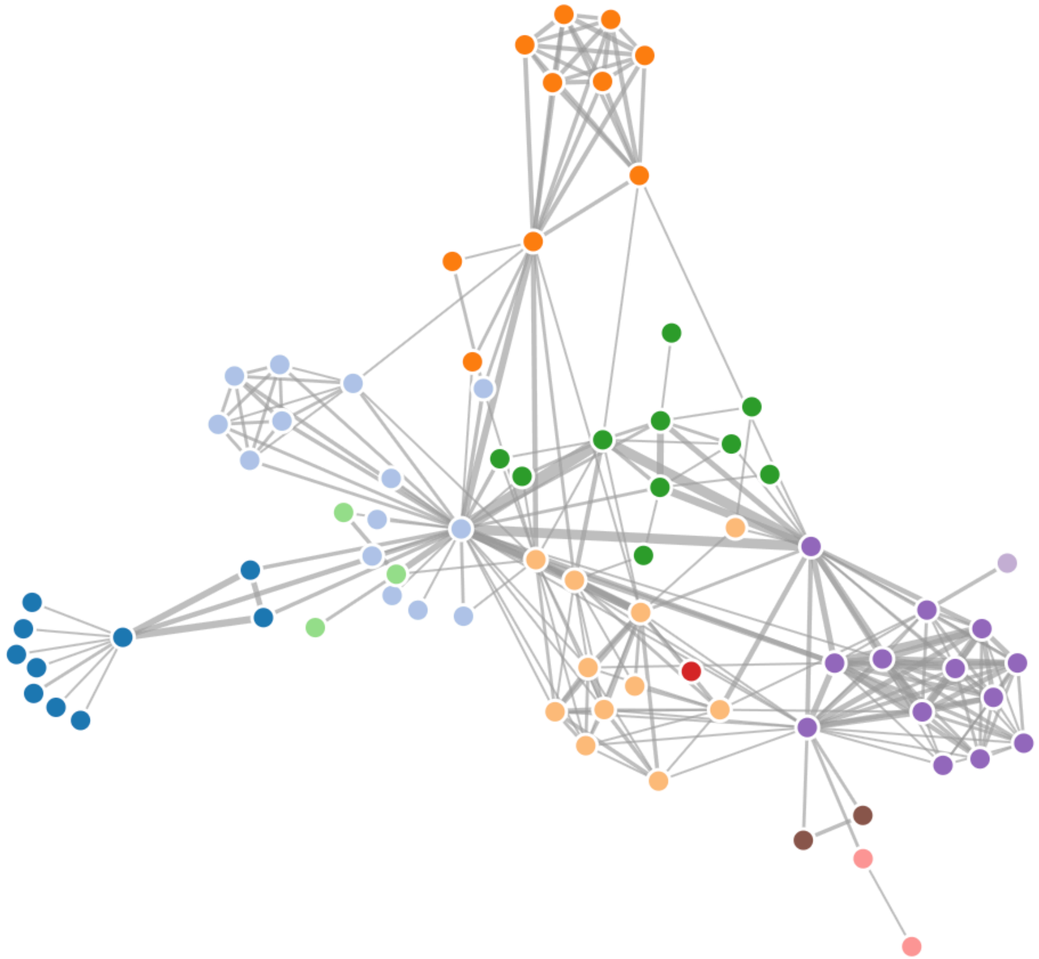
# Force-directed layouts

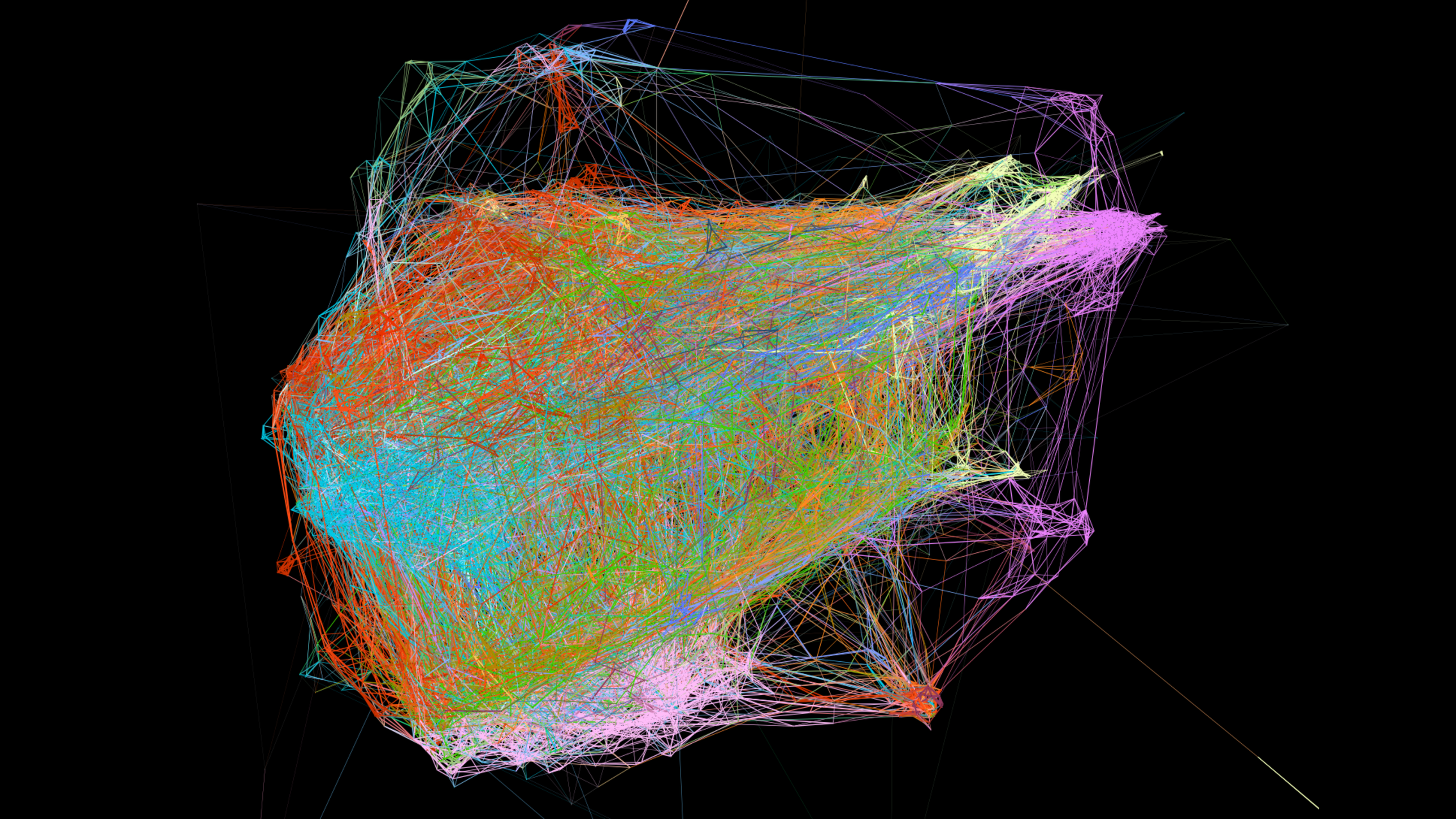
Most common layouts

Adapted from Physics, with vertices as repulsive magnets and edges as springs

Computationally expensive: O(n^3)

Hard scale for interactivity





# Node-link diagrams issues

Giant hairball depending on the topology of the graph

Computing an optimal layout lies in NP

Try to solve as an optimization problem, not necessarily convex

Complexity is still high O(n^2) - in each optimization step, all vertices are compared to all others

**[Lex]**

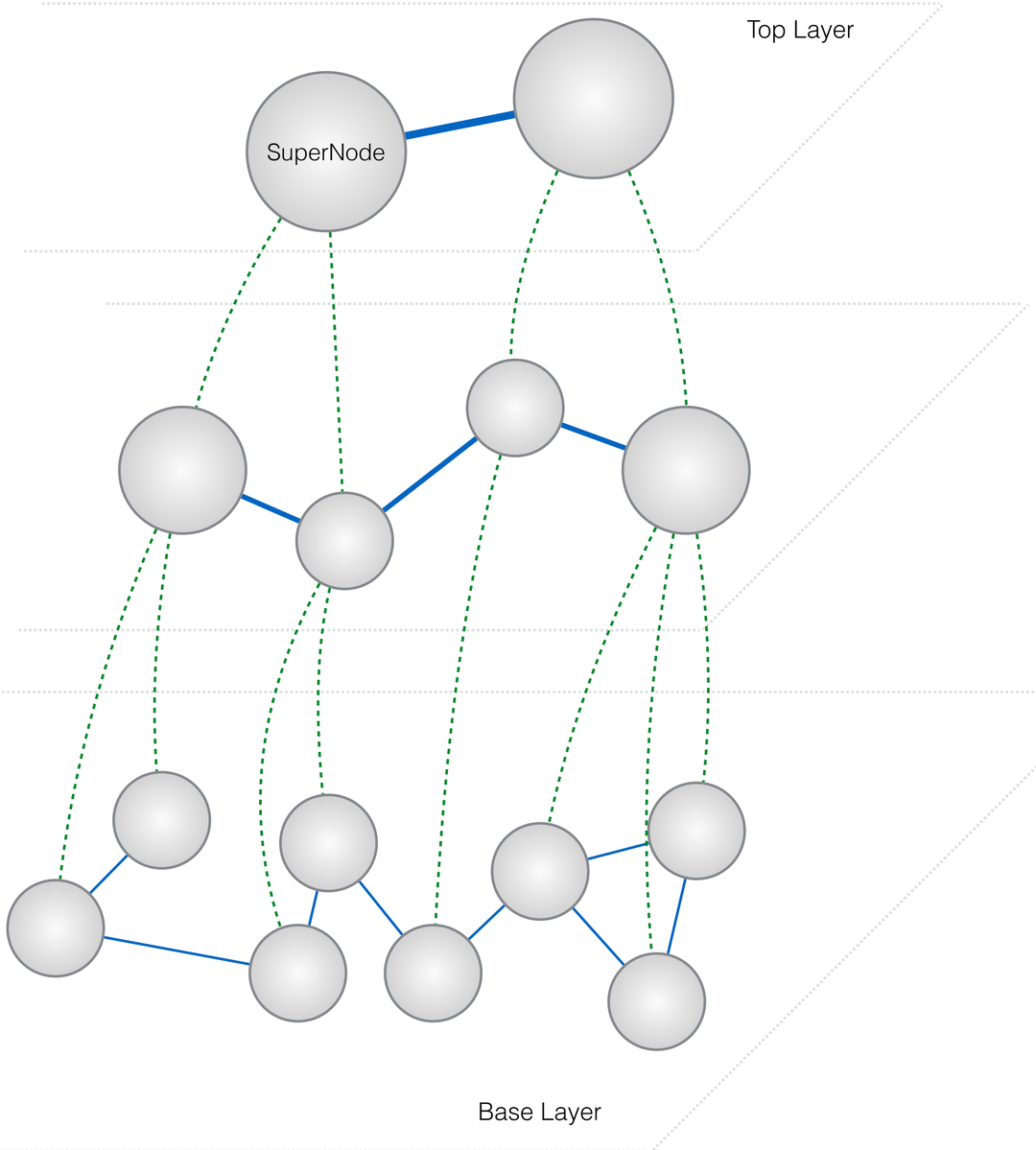
# Multilevel approaches

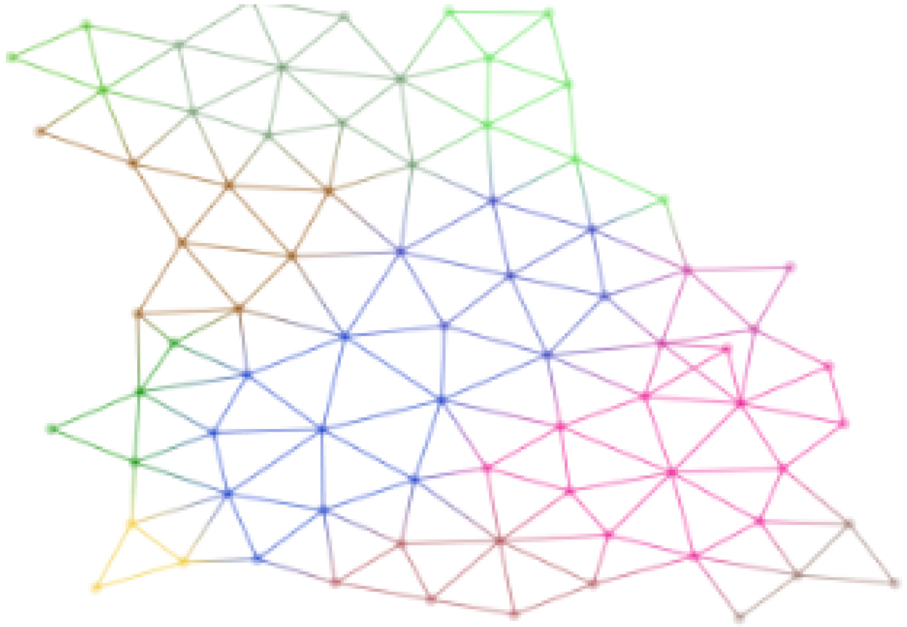
Also known as graph coarsening

Different strategies to collapse node or edges

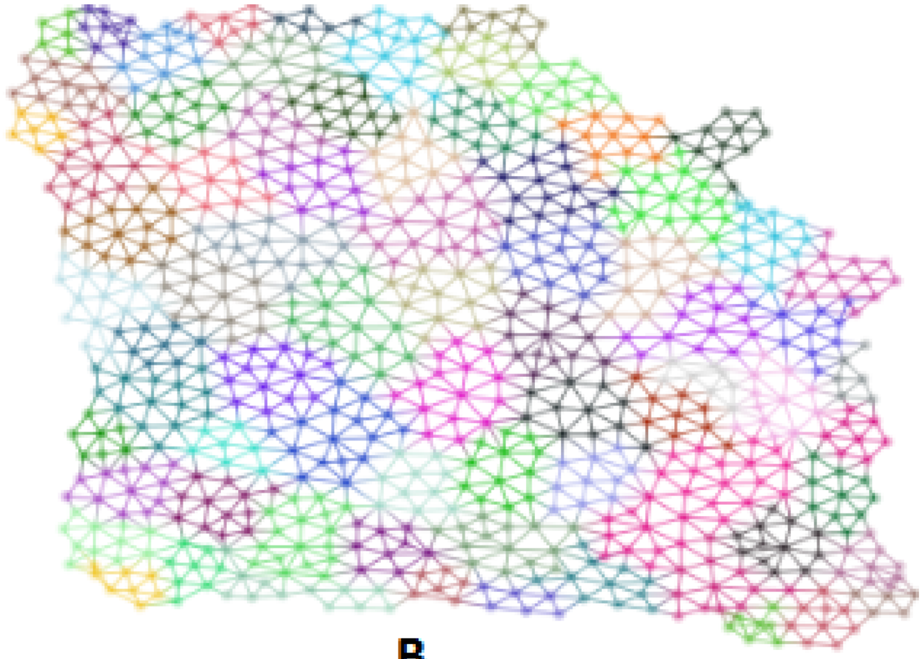
Recursive graph layout from top to bottom

Used also in graph partitioning, shortest path, etc.

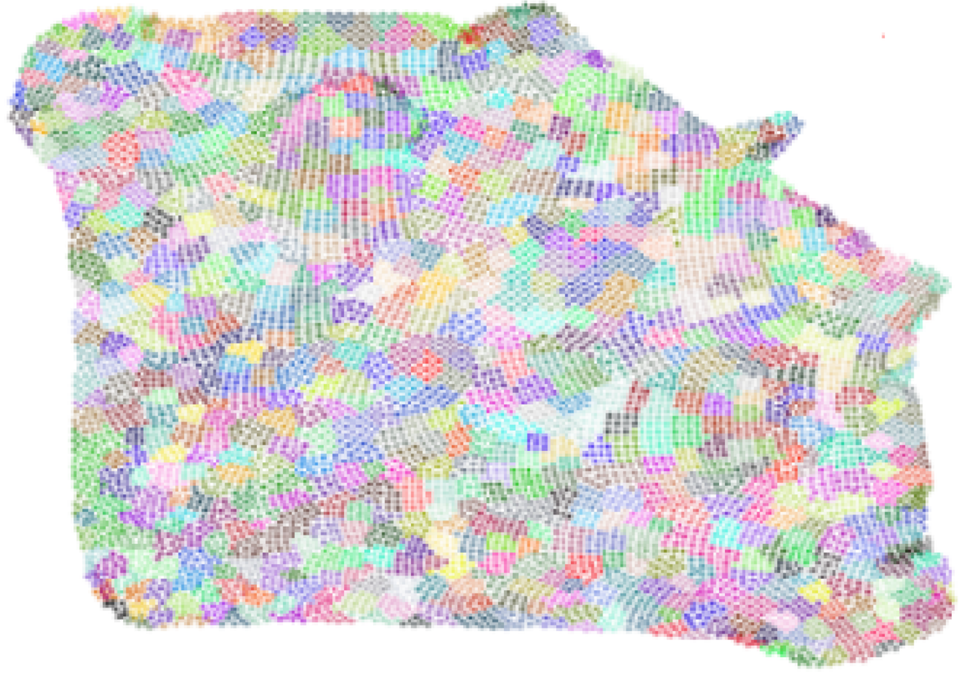




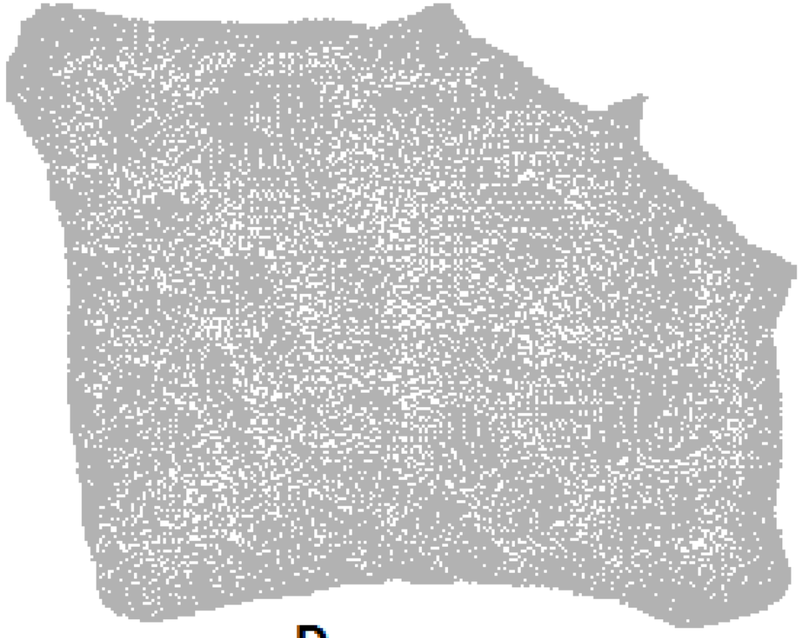
A



B



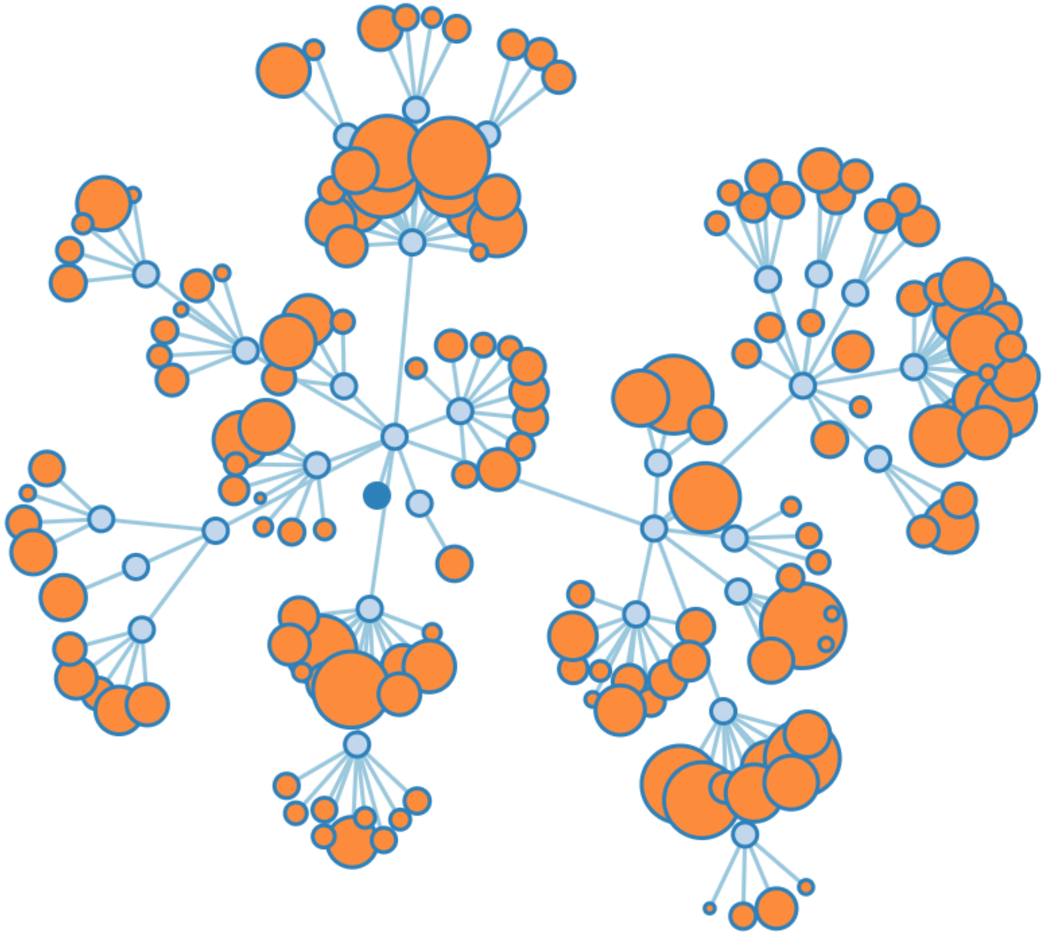
C



D

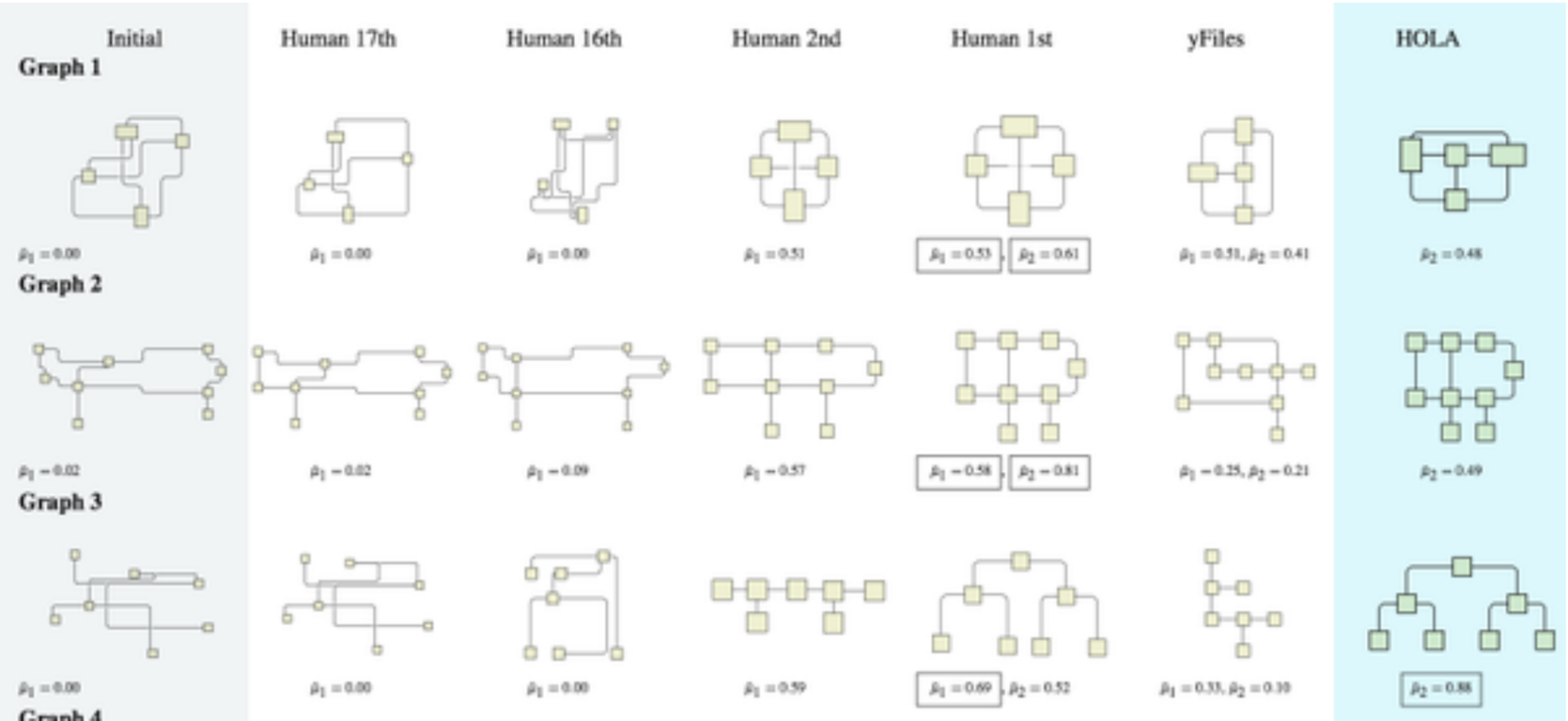
[Mi et al.]

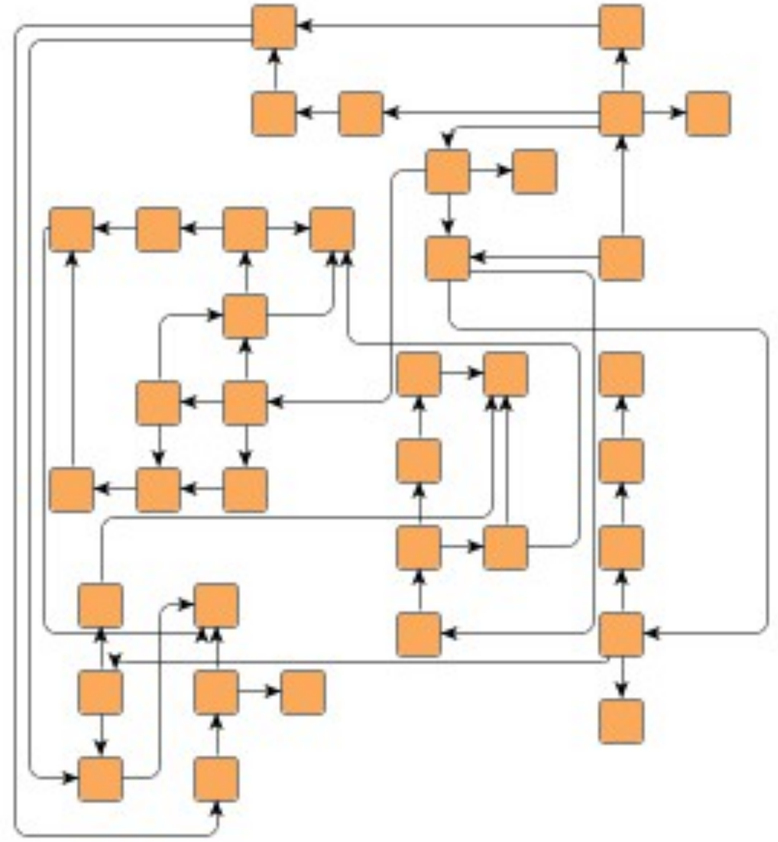
# Collapsible Force Layout



# Orthogonal layout

Used primarily in electrical engineering and software dev



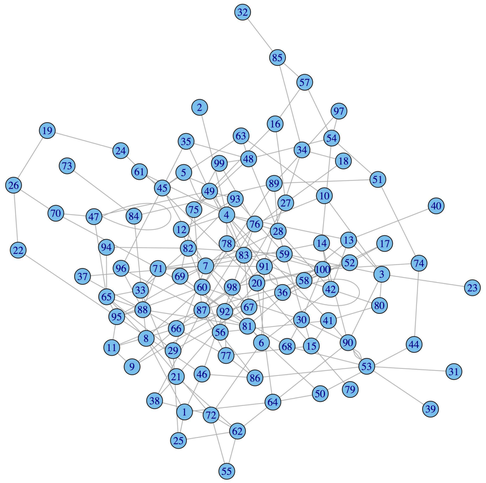



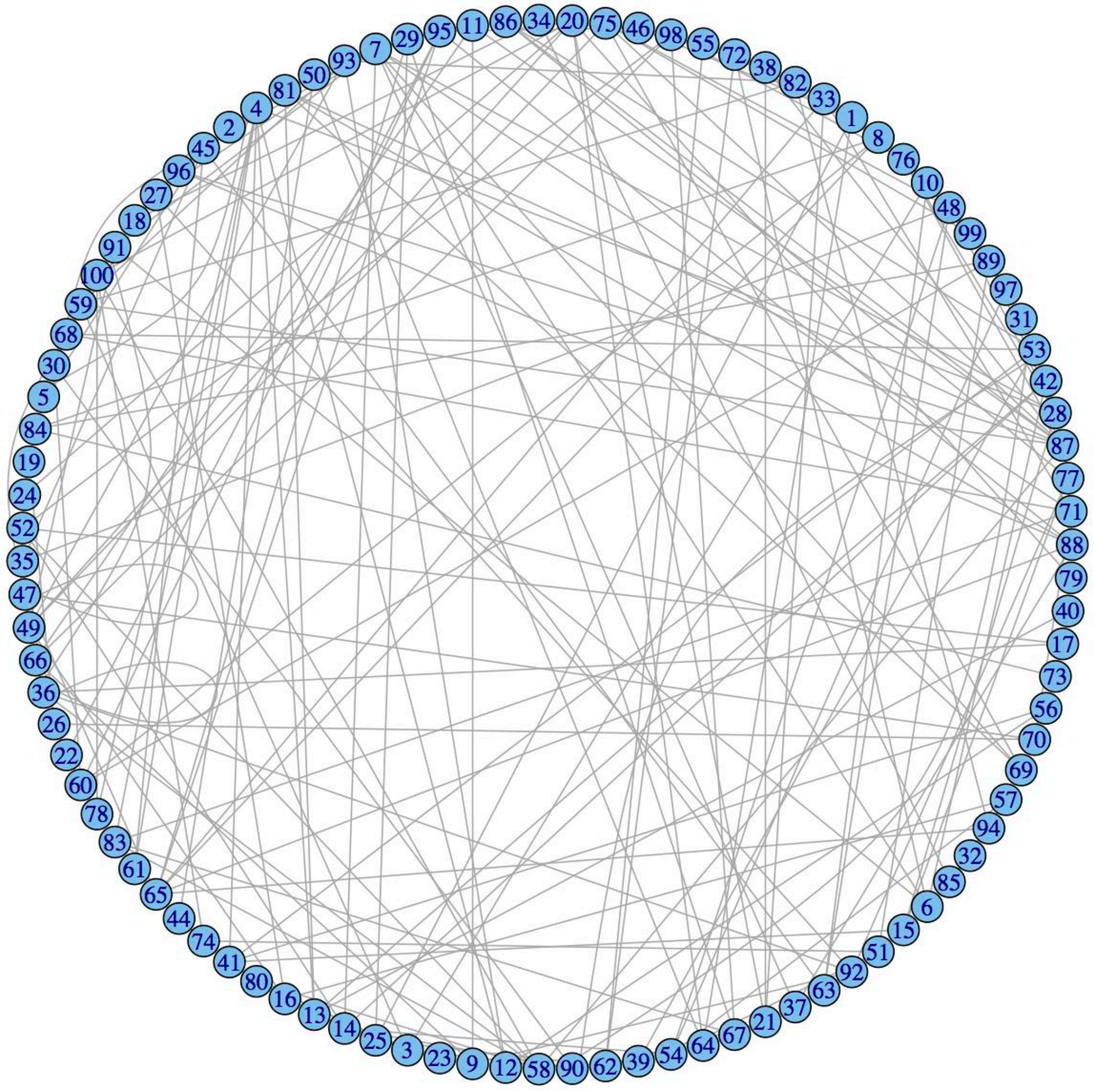
# Circle layout

Node positions are constrained on the edge of the circle

Clutter is important

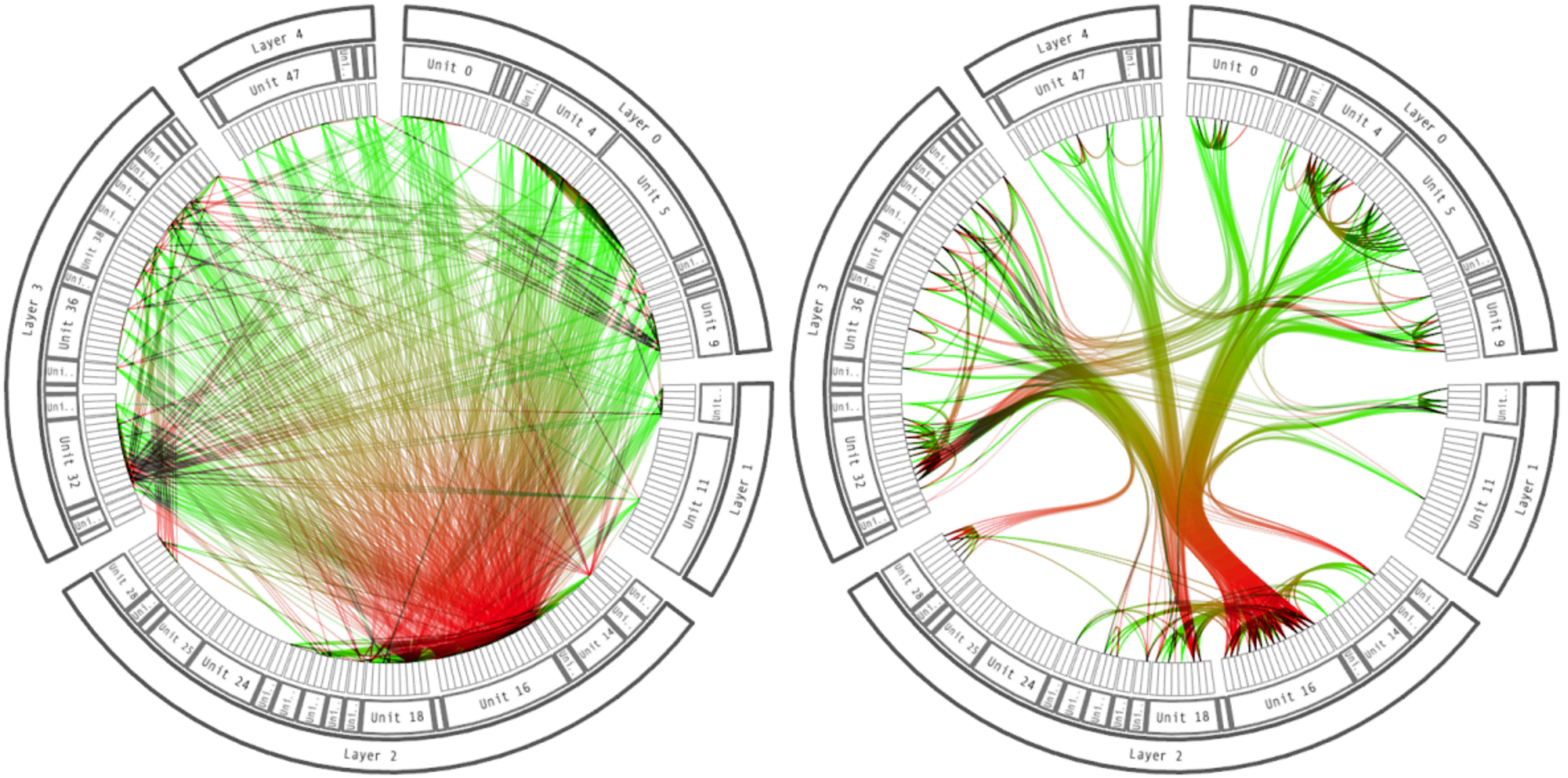
Node ordering is paramount



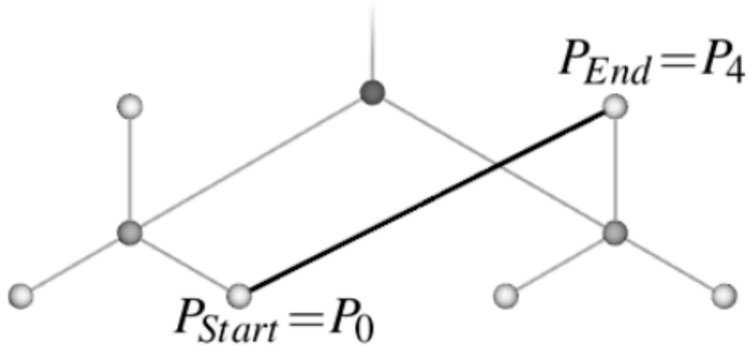


# Edge bundling

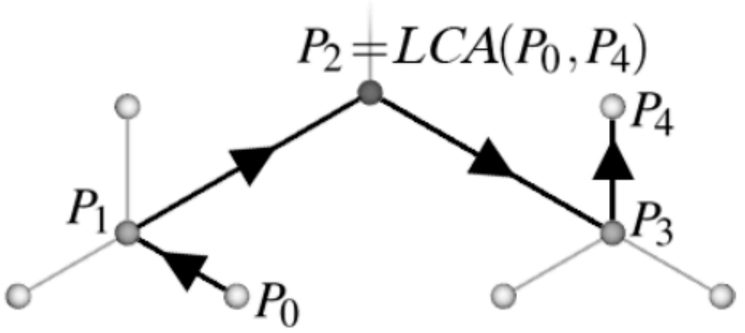
Reduce clutter by visually grouping edges



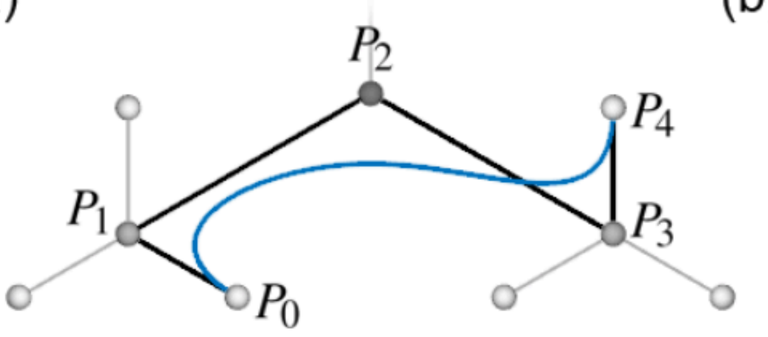

[Holten]



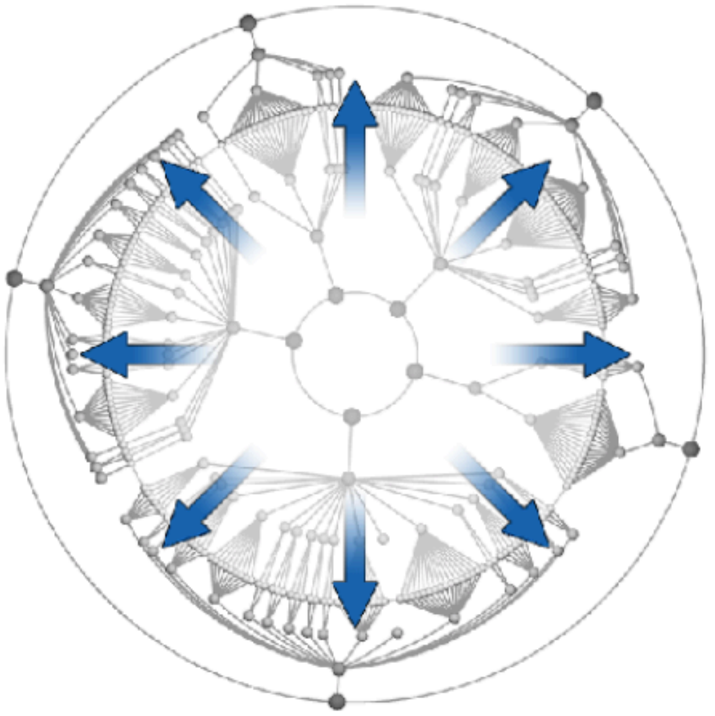

(a)

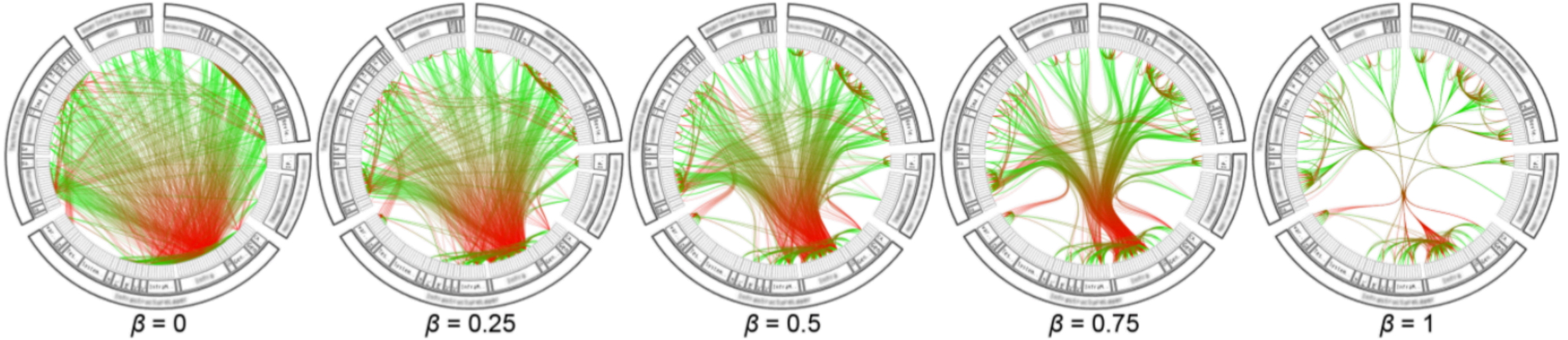



(b)

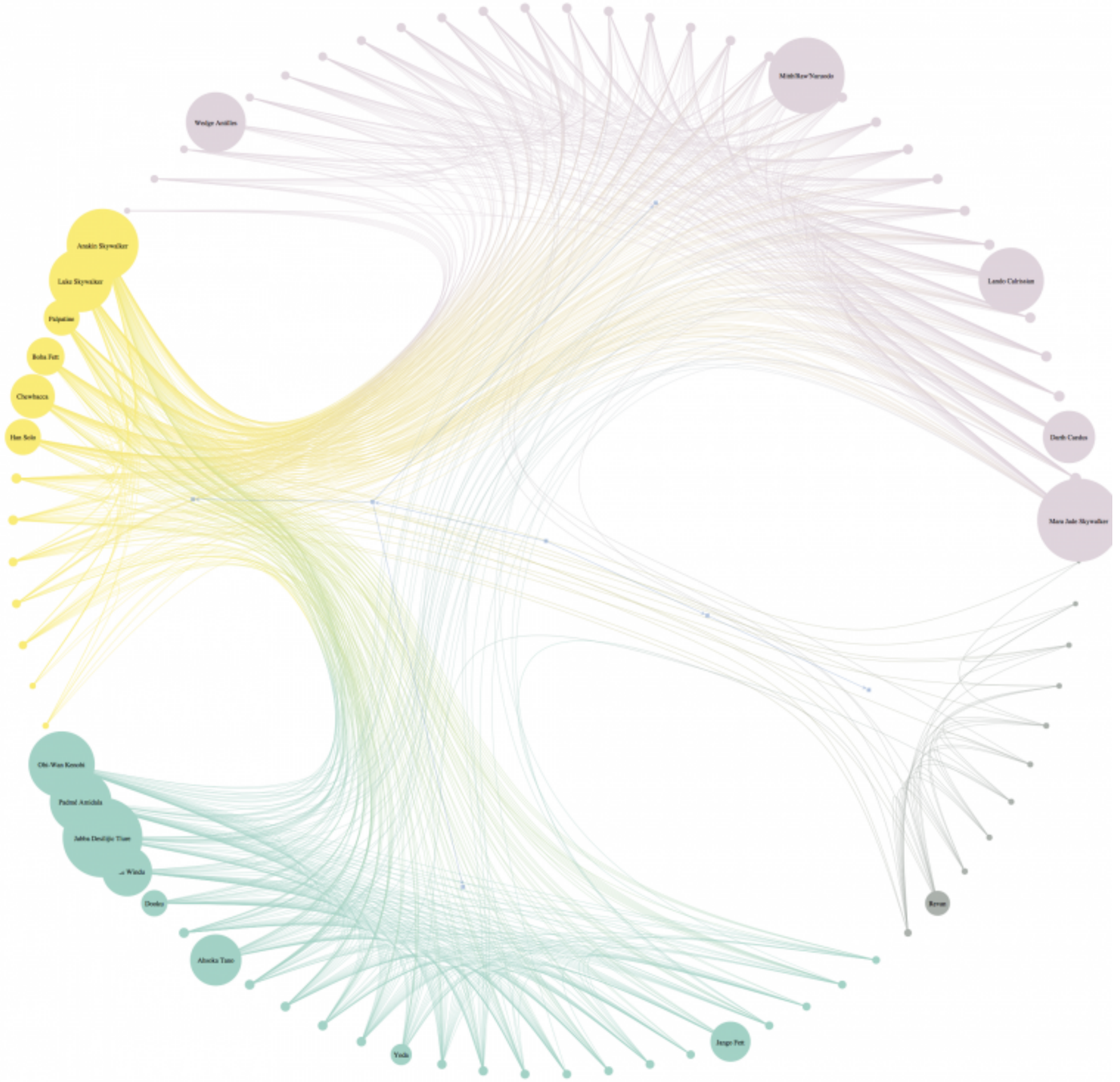


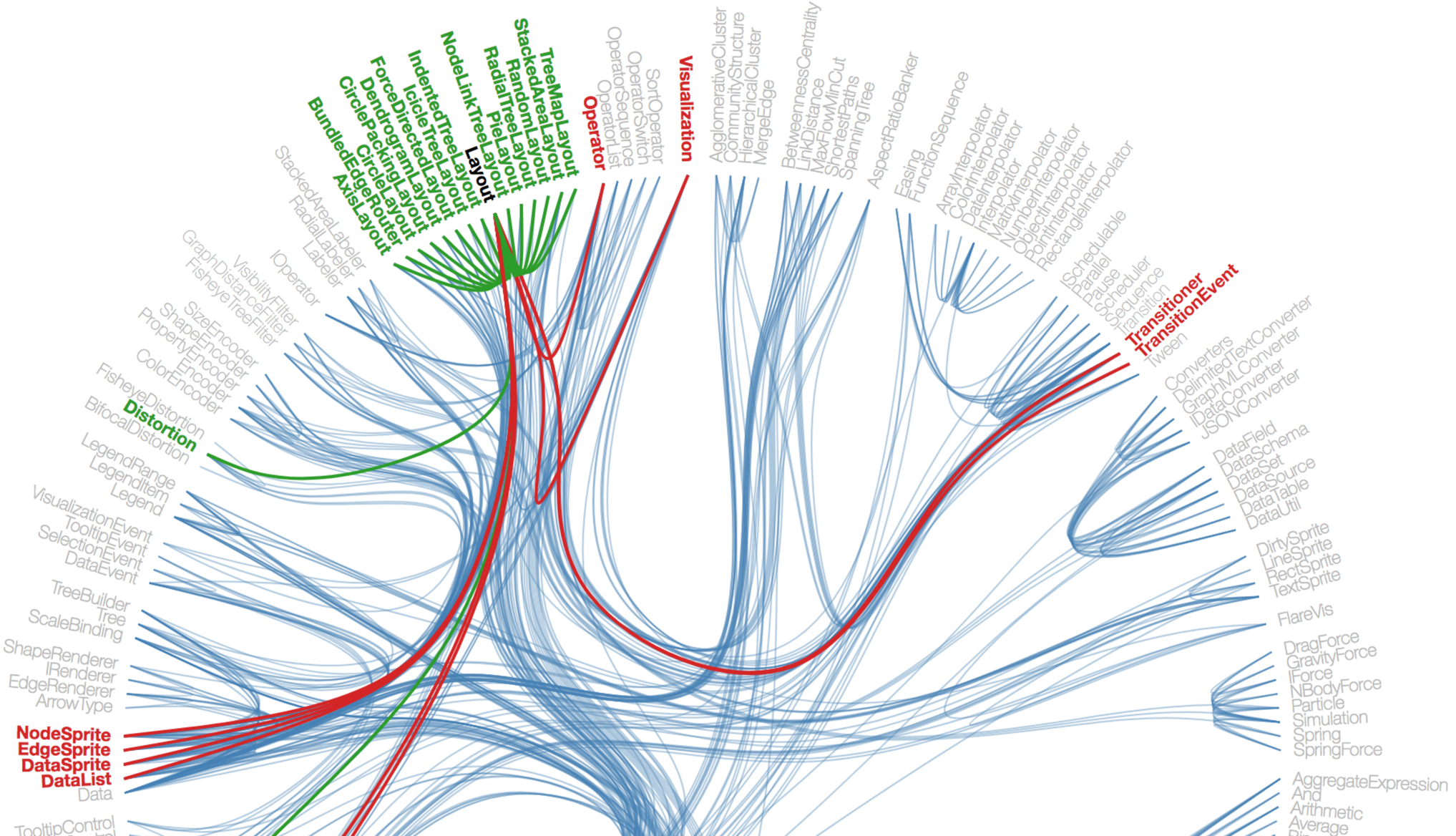

(c)



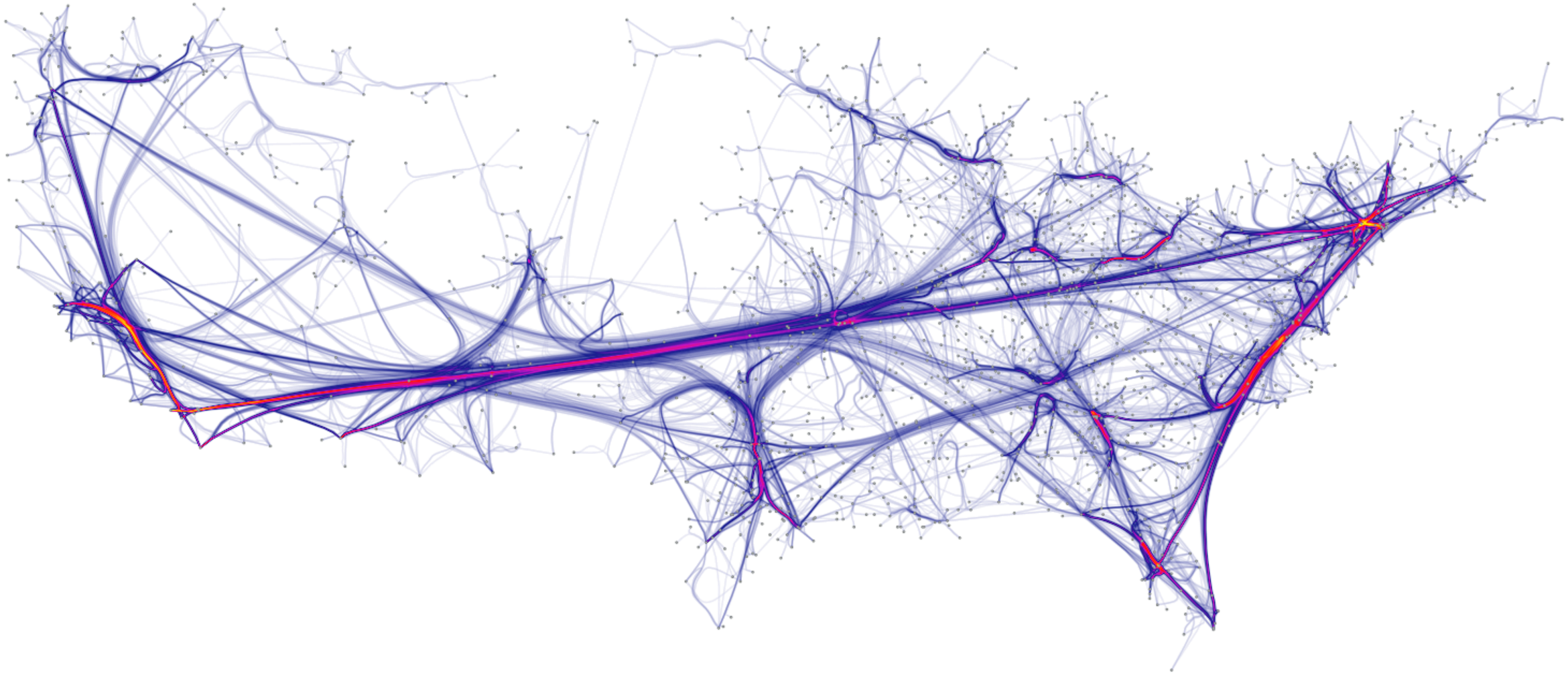



[Taken from Lex]


Mitth'Raw'Nuruodo
Wedge Antilles
Anakin Skywalker
Luke Skywalker
Palpatine
Boba Fett
Chewbacca
Han Solo
Lando Calrissian
Darth Caedus
Mara Jade Skywalker
Obi-Wan Kenobi
Padmé Amidala
Jaden Korr
Winda
Dooku
Ahsoka Tano
Yoda
Jango Fett
Revan


TreeMapLayout
StackedAreaLayout
RandomLayout
RadialTreeLayout
PieLayout
NodeLinkTreeLayout
Layout
IndentedTreeLayout
IcicleTreeLayout
ForceDirectedLayout
DendrogramLayout
CirclePackingLayout
CircleLayout
BundledEdgeRouter
AxisLayout
Visualization
Operator
OpSort
OpOperatorSequence
OpOperatorSwitch
SortOperator
OperatorSequence
OperatorSwitch
OperatorList
AgglomerativeCluster
CommunityStructure
HierarchicalCluster
MergeEdge
BetweennessCentrality
LinkDistance
MaxFlowMinCut
ShortestPaths
SpanningTree
AspectRatioBanker
Easing
FunctionSequence
ArrayInterpolator
ColorInterpolator
DateInterpolator
Interpolator
MatrixInterpolator
NumberInterpolator
ObjectInterpolator
PointInterpolator
RectangleInterpolator
ISchedulable
Parallel
Pause
Scheduler
Sequence
Transition
Transitioner
TransitionEvent
Tween
StackedAreaLabeler
RadialLabeler
Labeler
IOperator
VisibilityFilter
GraphDistanceFilter
FisheyeTreeFilter
SizeEncoder
ShapeEncoder
PropertyEncoder
Encoder
ColorEncoder
FisheyeDistortion
BifocalDistortion
Distortion
LegendRange
LegendItem
Legend
VisualizationEvent
TooltipEvent
SelectionEvent
DataEvent
TreeBuilder
Tree
ScaleBinding
ShapeRenderer
IRenderer
EdgeRenderer
ArrowType
NodeSprite
EdgeSprite
DataSprite
DataList
Data
TooltipControl
Converters
DelimitedTextConverter
GraphMLConverter
IDataConverter
JSONConverter
DataField
DataSchema
DataSet
DataSource
DataTable
DataUtil
DirtySprite
LineSprite
RectSprite
TextSprite
FlareVis
DragForce
GravityForce
IForce
NBodyForce
Particle
Simulation
Spring
SpringForce
AggregateExpression
And
Arithmetic
Average

# Fixed layout



[Holten]


facebook
December 2010

# Node-link diagrams

**Pros**

Visualize all graph classes

Very flexible in the layout

Good for topology related queries (if good layout)

**Cons**

Tendency to clutter

Hard to find a good layout (lies in NP)

Even heuristics are still slow/complex to run

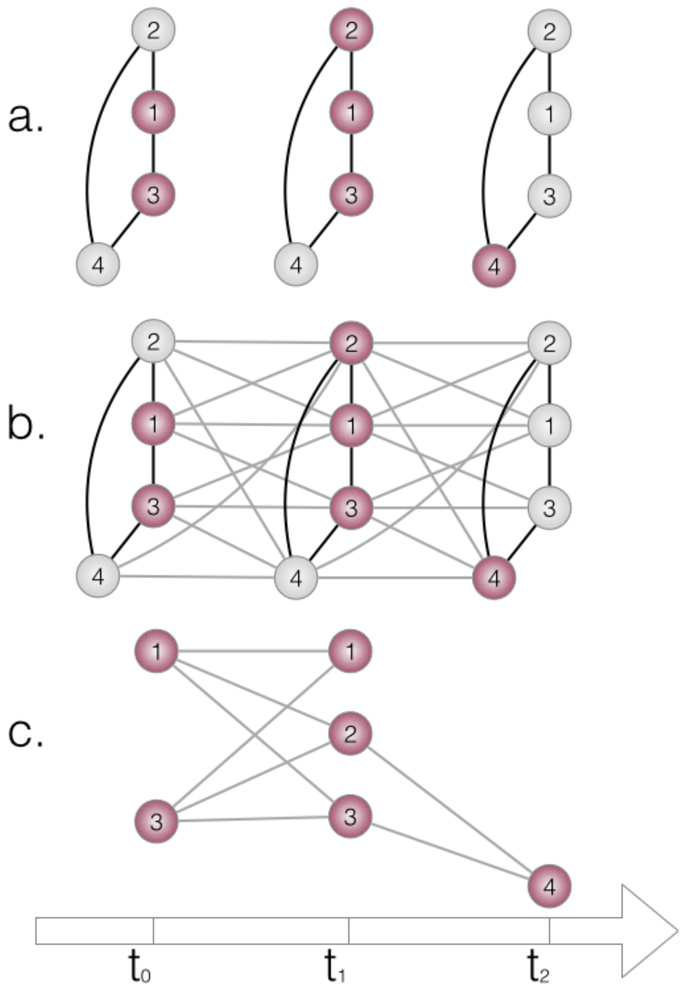
# Multivariate graphs

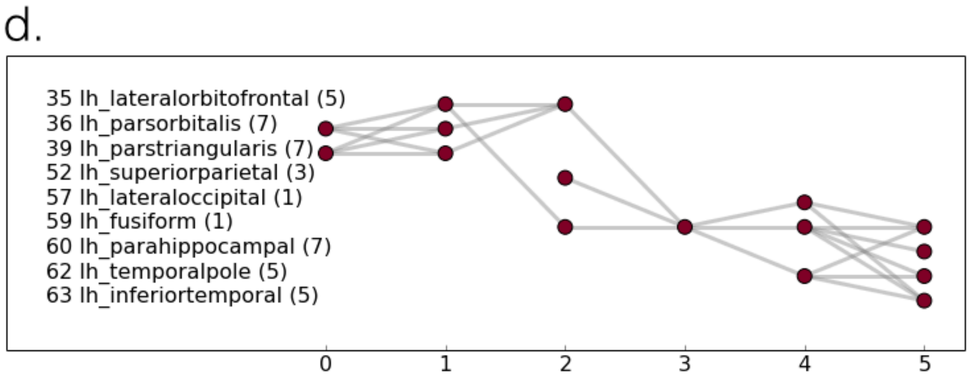
# Multivariate Graphs

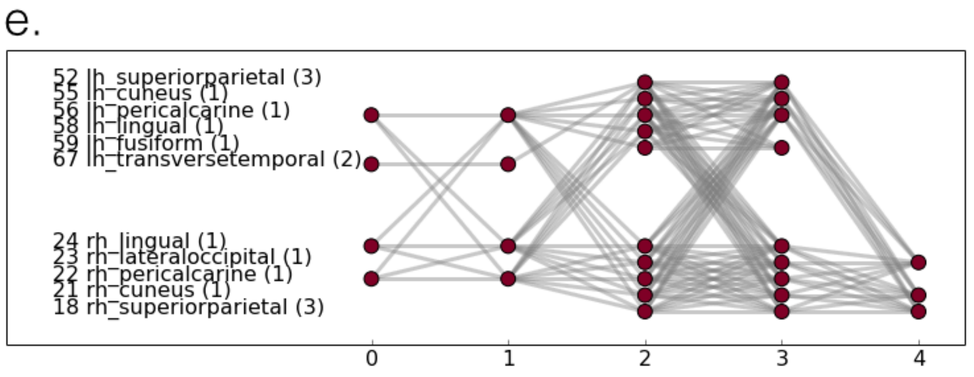
How to represent both the topology and attributes on the graph?

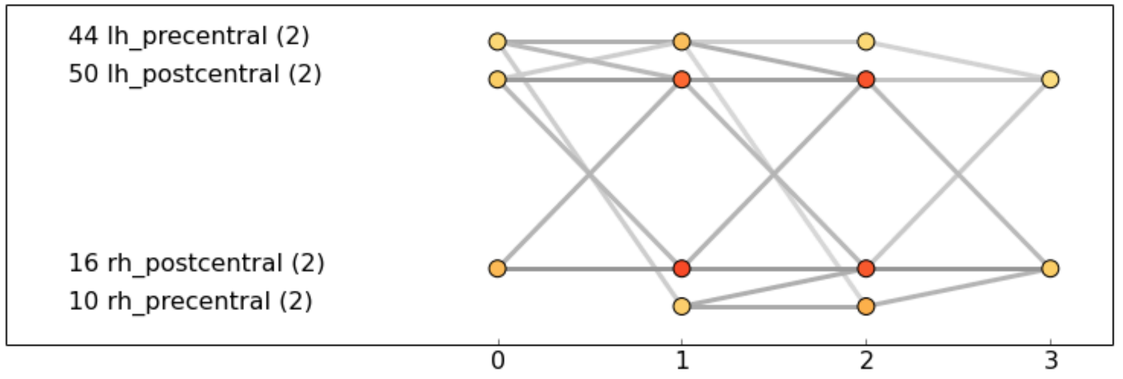
Attributes can also influence topology

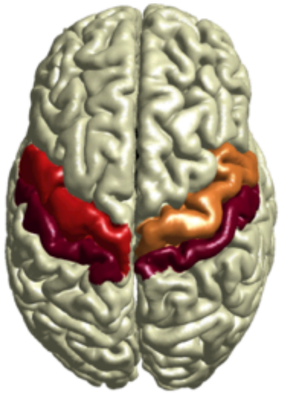
Show/hide edges or paths


a.
b.
c.
t0
t1
t2


d.
35 lh_lateralorbitofrontal (5)
36 lh_parsorbitalis (7)
39 lh_parstriangularis (7)
52 lh_superiorparietal (3)
57 lh_lateraloccipital (1)
59 lh_fusiform (1)
60 lh_parahippocampal (7)
62 lh_temporalpole (5)
63 lh_inferiortemporal (5)
0
1
2
3
4
5


e.
52 lh_superiorparietal (3)
55 lh_cuneus (1)
56 lh_pericalcarine (1)
58 lh_lingual (1)
59 lh_fusiform (1)
67 lh_transversetemporal (2)
24 rh_lingual (1)
23 rh_lateraloccipital (1)
22 rh_pericalcarine (1)
21 rh_cuneus (1)
18 rh_superiorparietal (3)
0
1
2
3
4

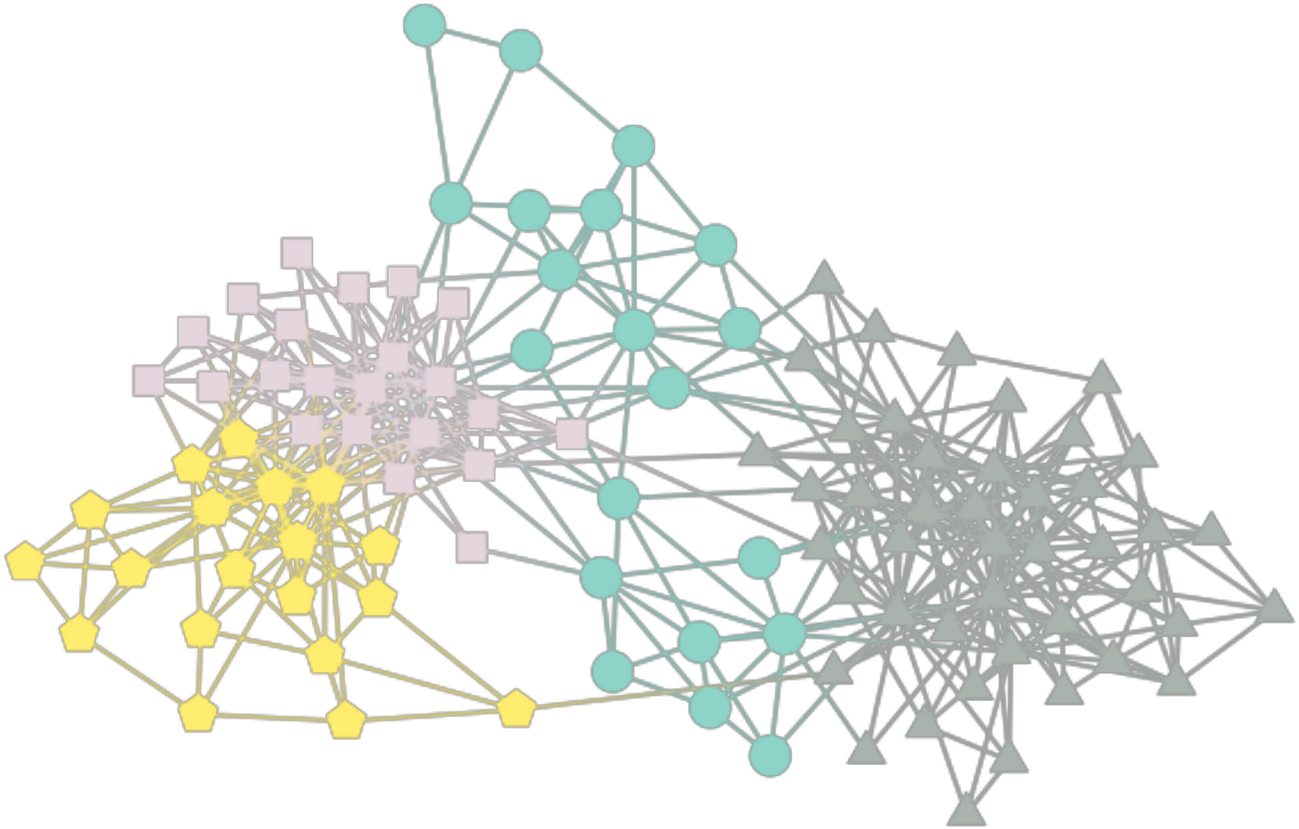

44 lh_precentral (2)
50 lh_postcentral (2)
16 rh_postcentral (2)
10 rh_precentral (2)
0
1
2
3



# Node attributes

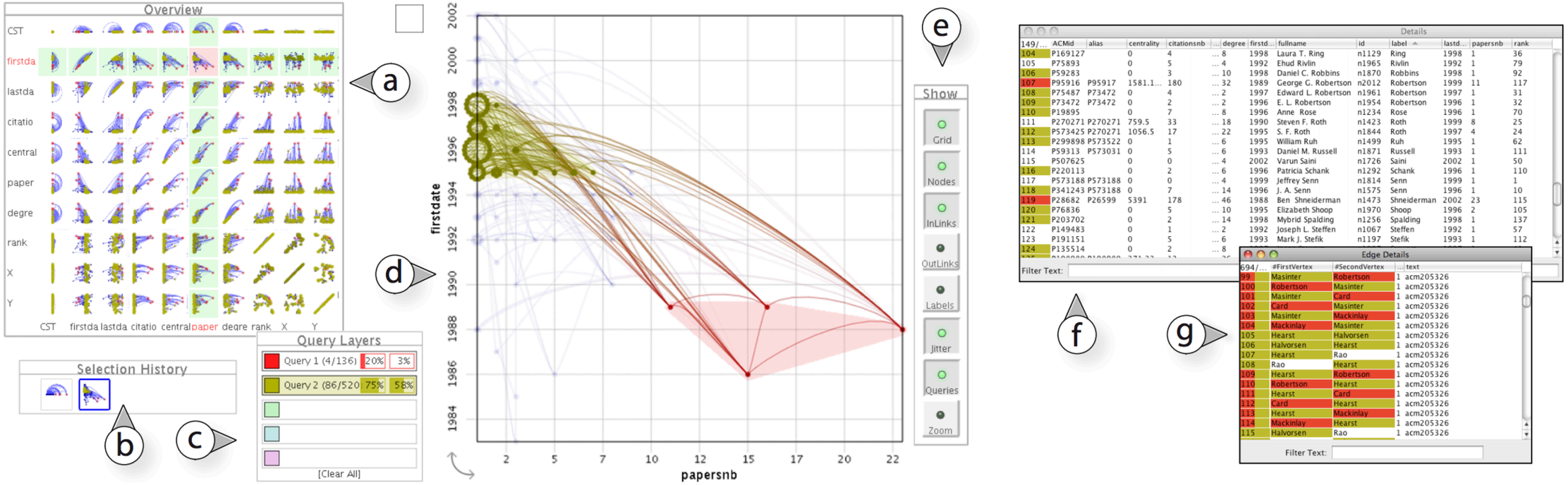
Natural choice is color or glyphs

Limited in scalability



# GraphDice

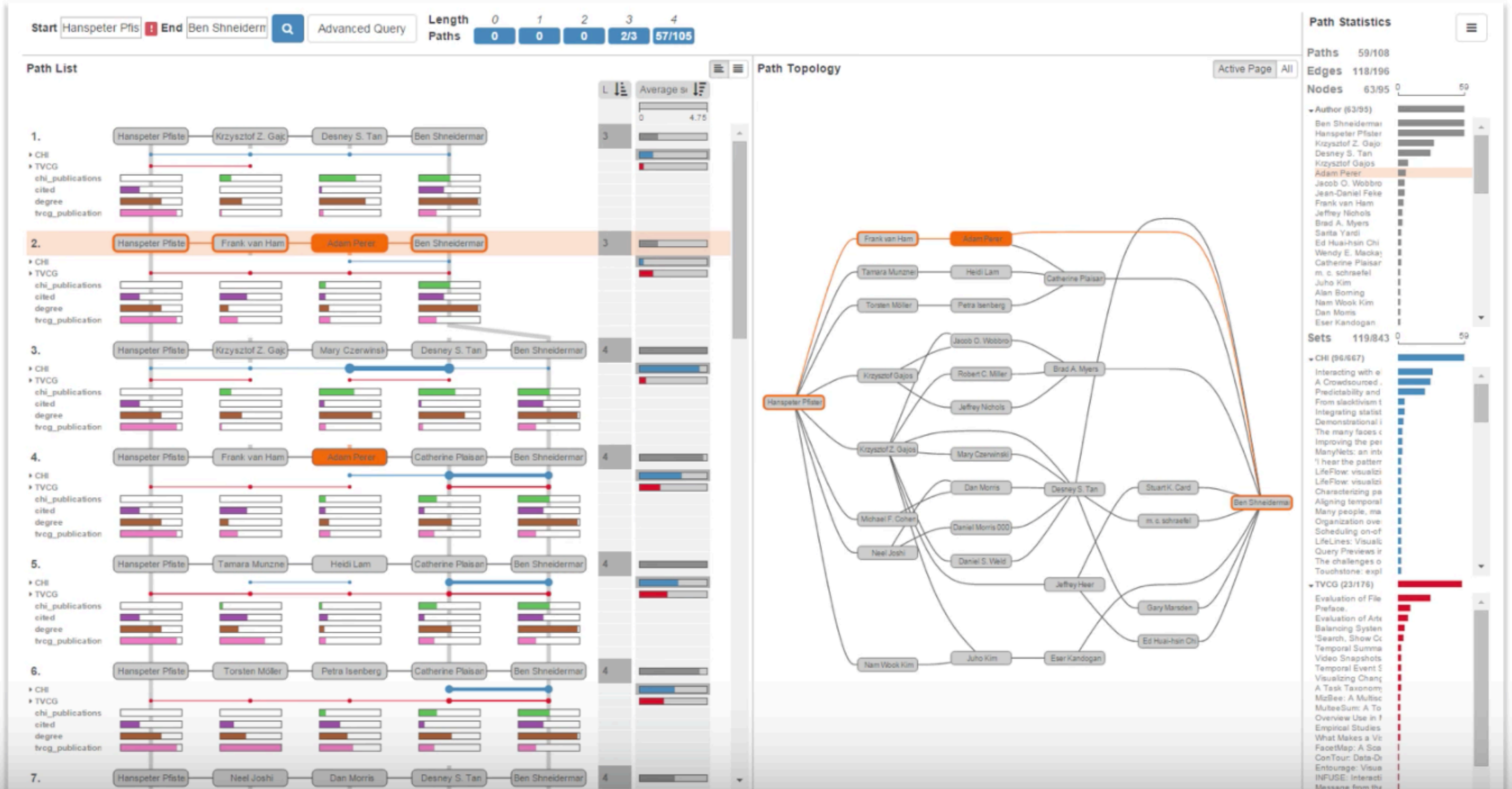
Nodes are positionned in space according to attribute values




[Bezerianos et al, 2010]

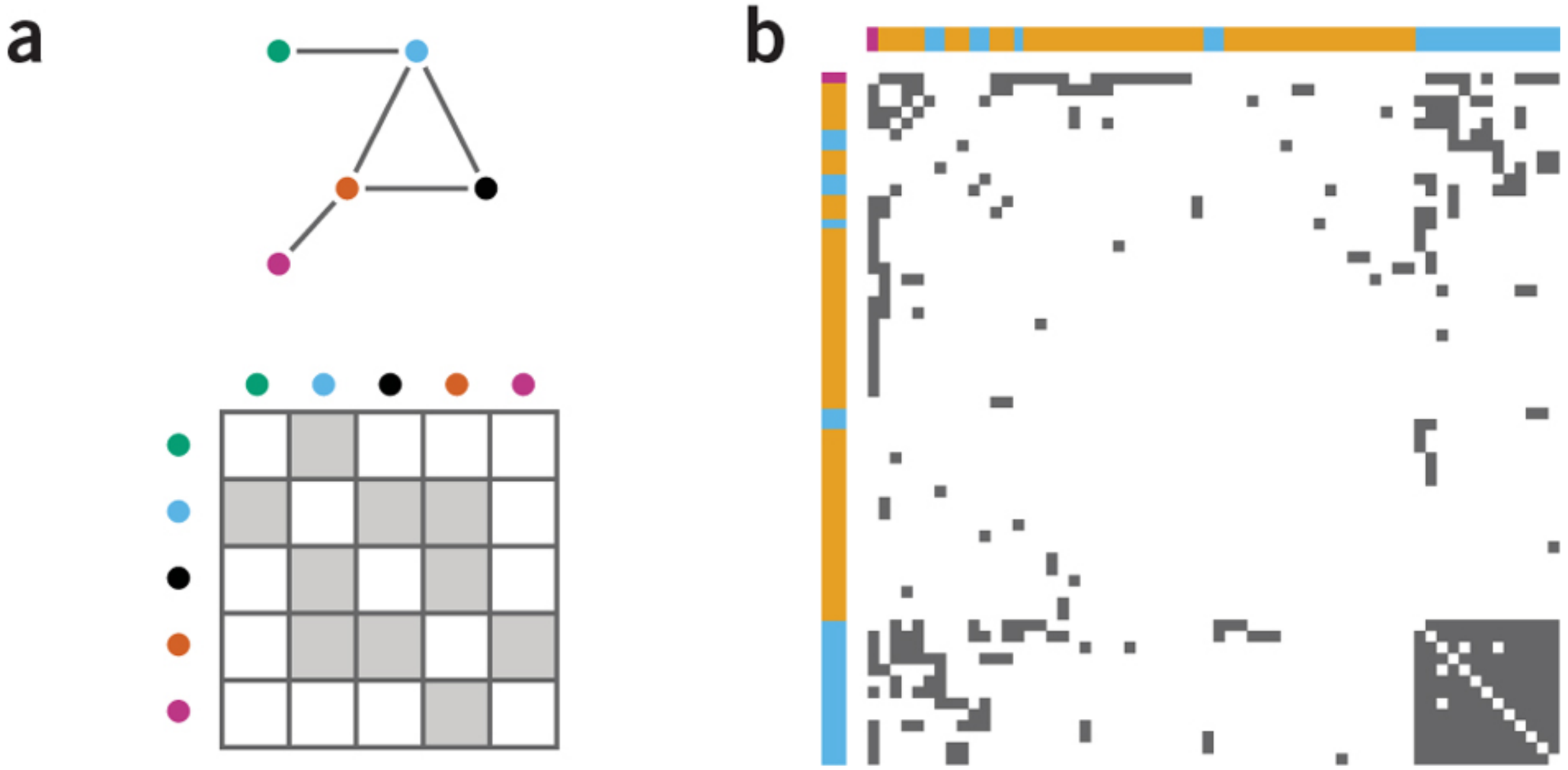
# PathFinder

[Partl et al.]

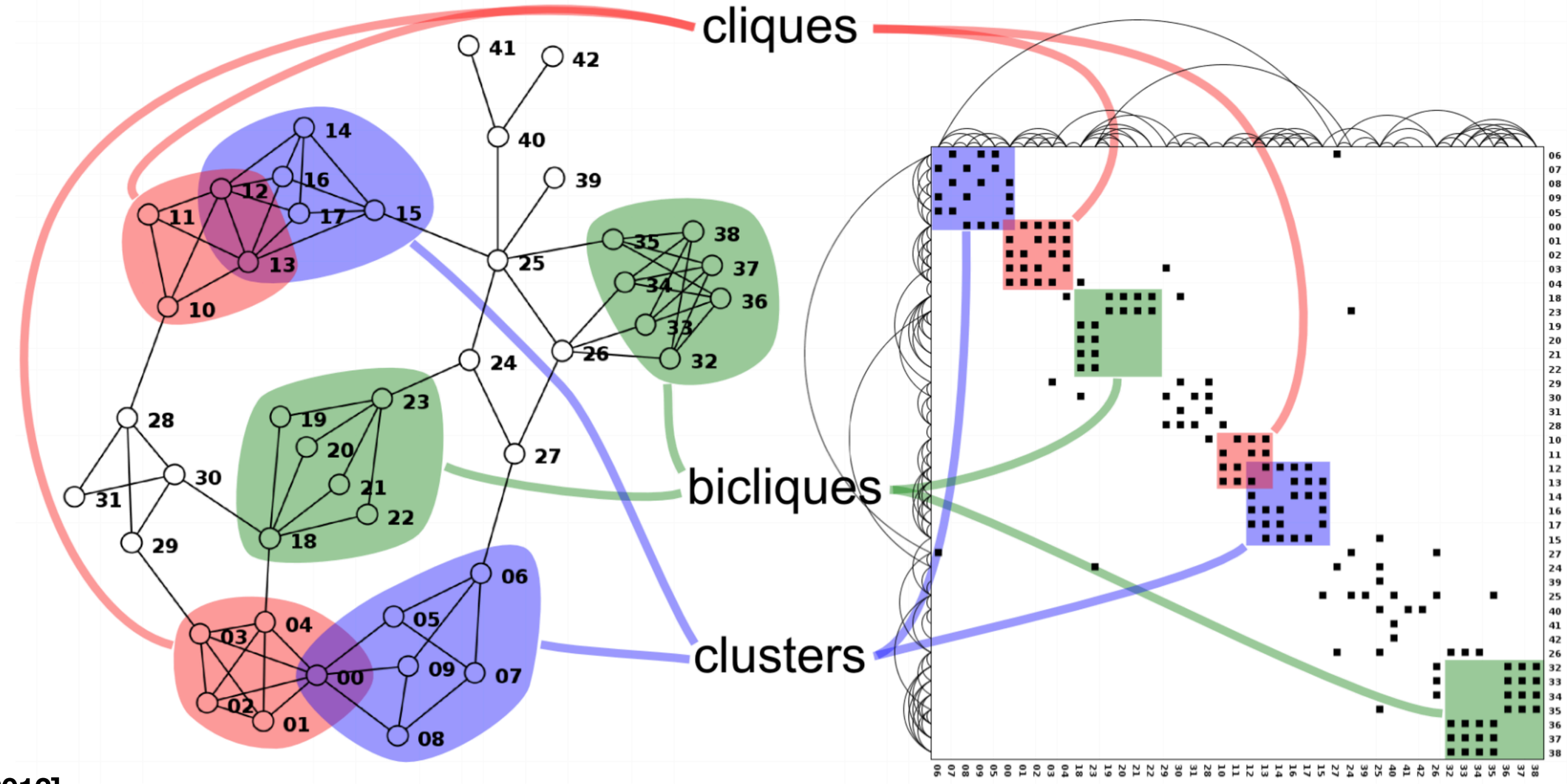


# Matrix representation

# Matrix representation

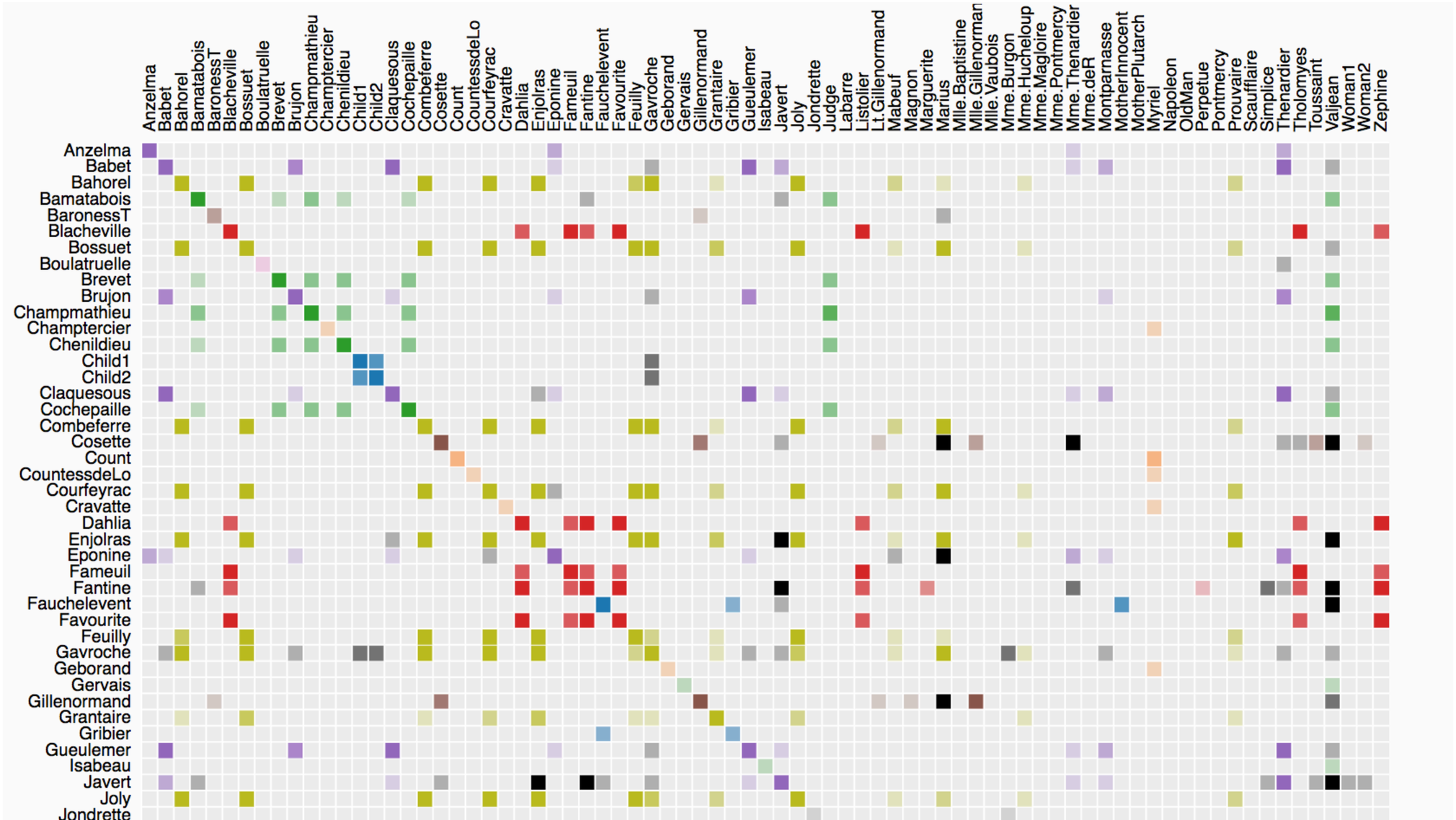


# Good for neighborhood tasks

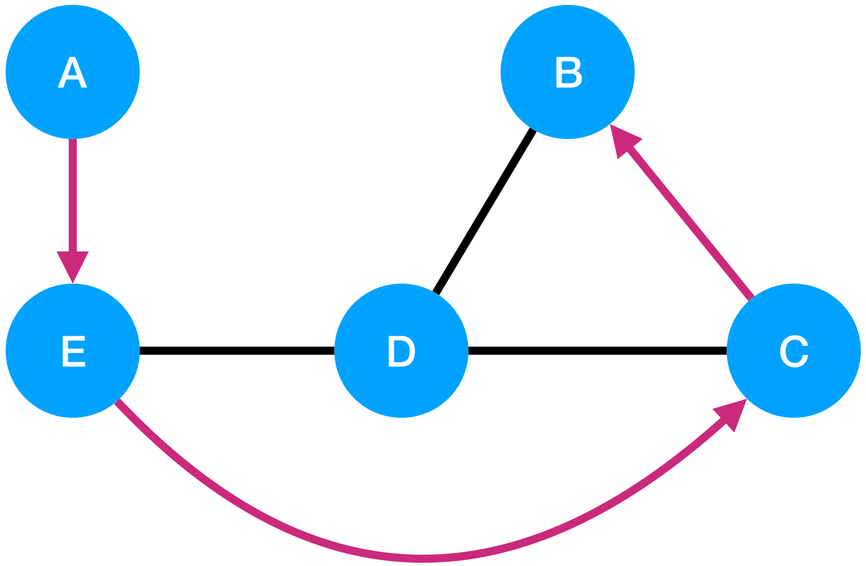



[McGuffin 2012]

# Node ordering is paramount!



# Don’t use for paths!



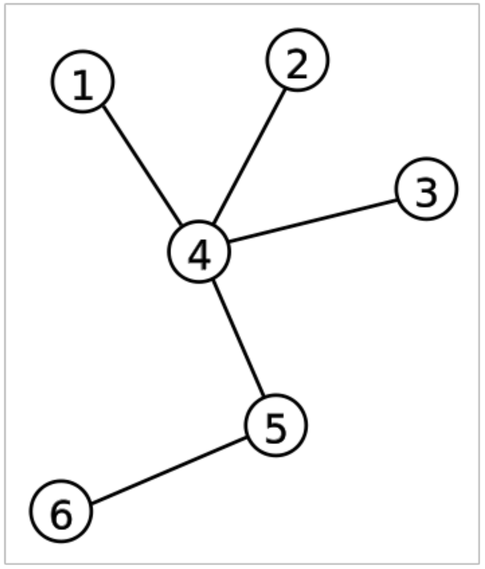

| | A | B | C | D | E |
|---|---|---|---|---|---|
| A | | | | | 1 |
| B | | | | | |
| C | | 3 | | | |
| D | | | | | |
| E | | | 2 | | |

| | Matrix-based representations | Node-link diagrams |
|---|---|---|
| + | No node overlapping<br>No edge crossing<br>Readable for dense graphs<br>Fast navigation<br>Fast manipulation<br>More readable for some tasks | Intuitive<br>Compact<br>More readable for path following<br>More effective for small graphs<br>More effective for sparse graphs |
| – | Less intuitive<br>Use more space<br>Weak for path following tasks | Useless without layout<br>Node overlapping<br>Edge crossing<br>Not readable for dense graphs<br>Manipulation requires layout computation |

[Heymann 2014]

# Tree visualization

# Trees

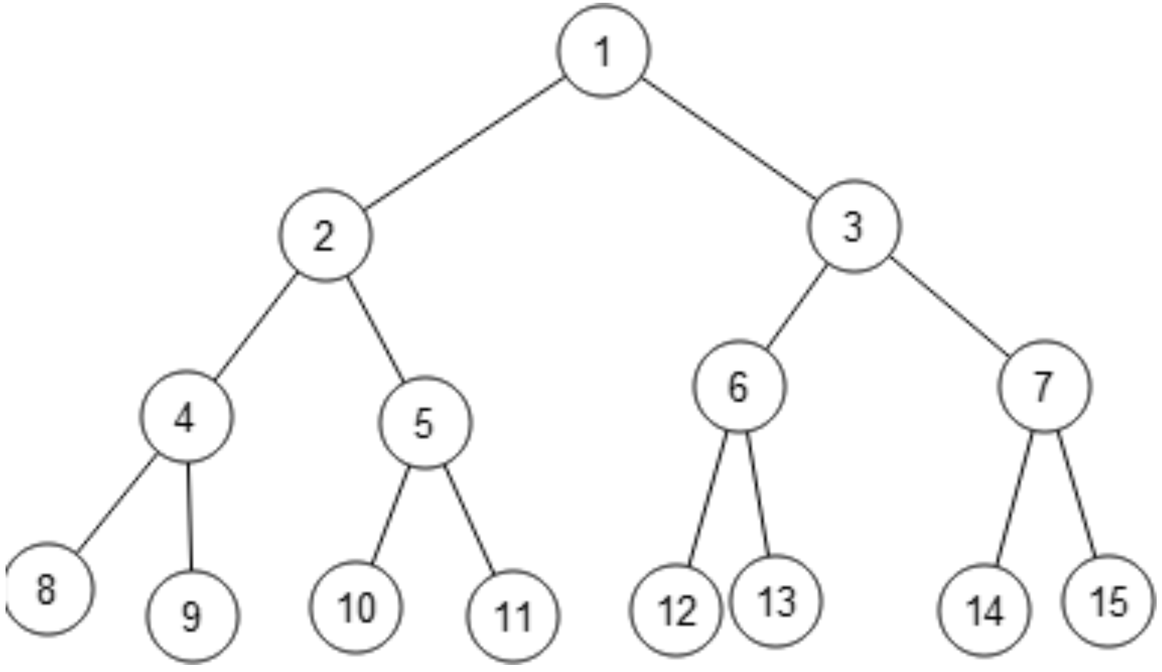



A tree is a graph constraints:

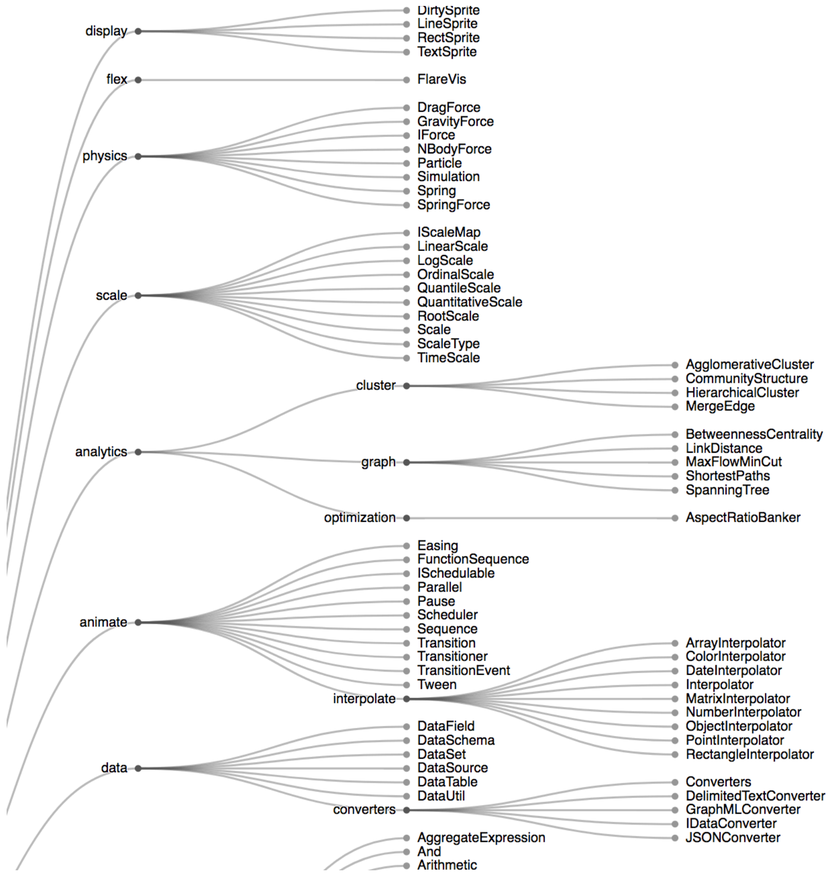
no cycles

can have a root node and subtrees

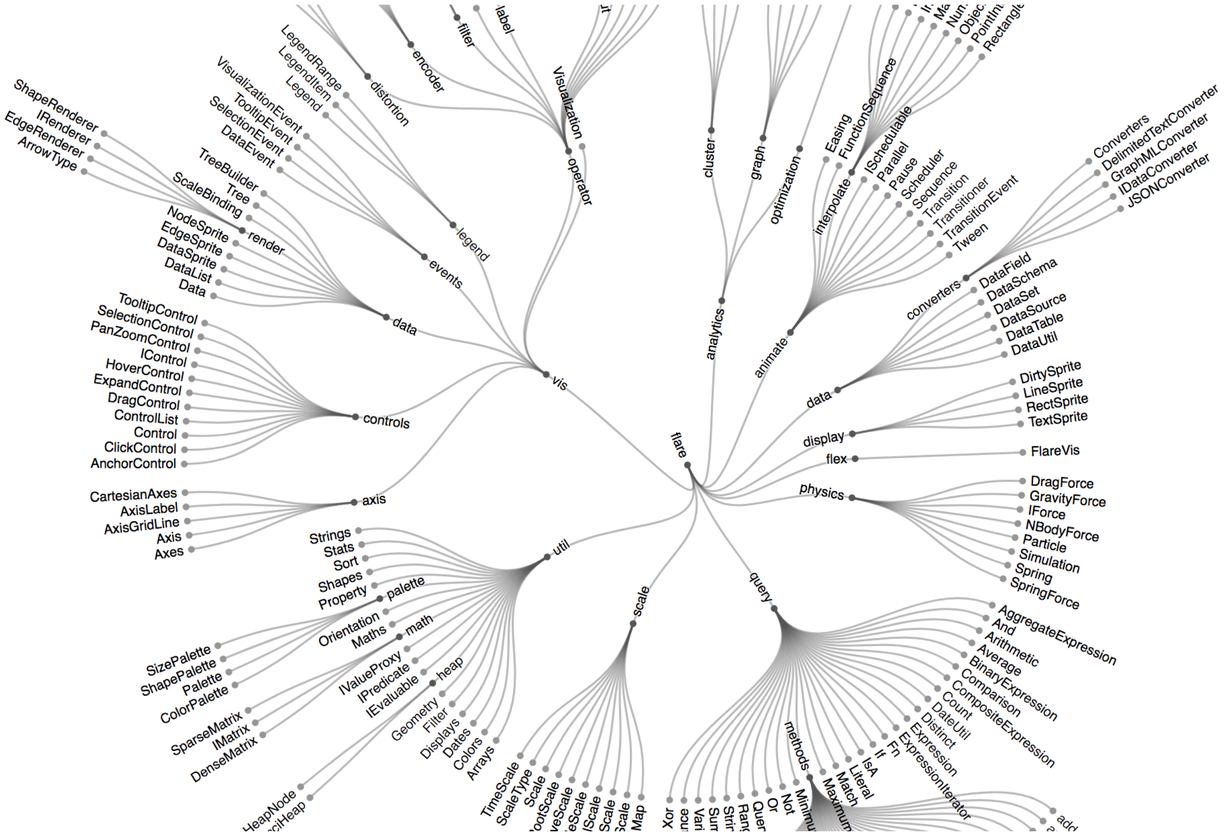
can be ordered



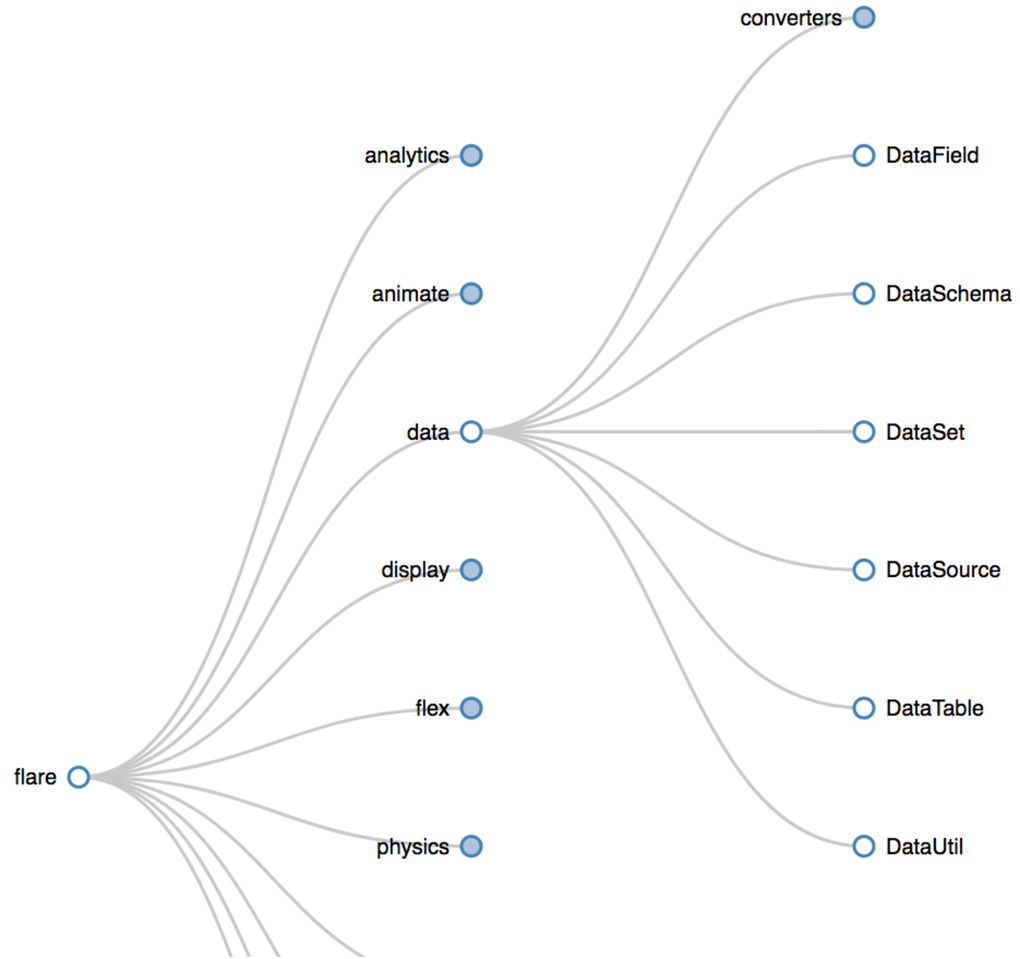
# Node-link diagram




## Reingold–Tilford layout

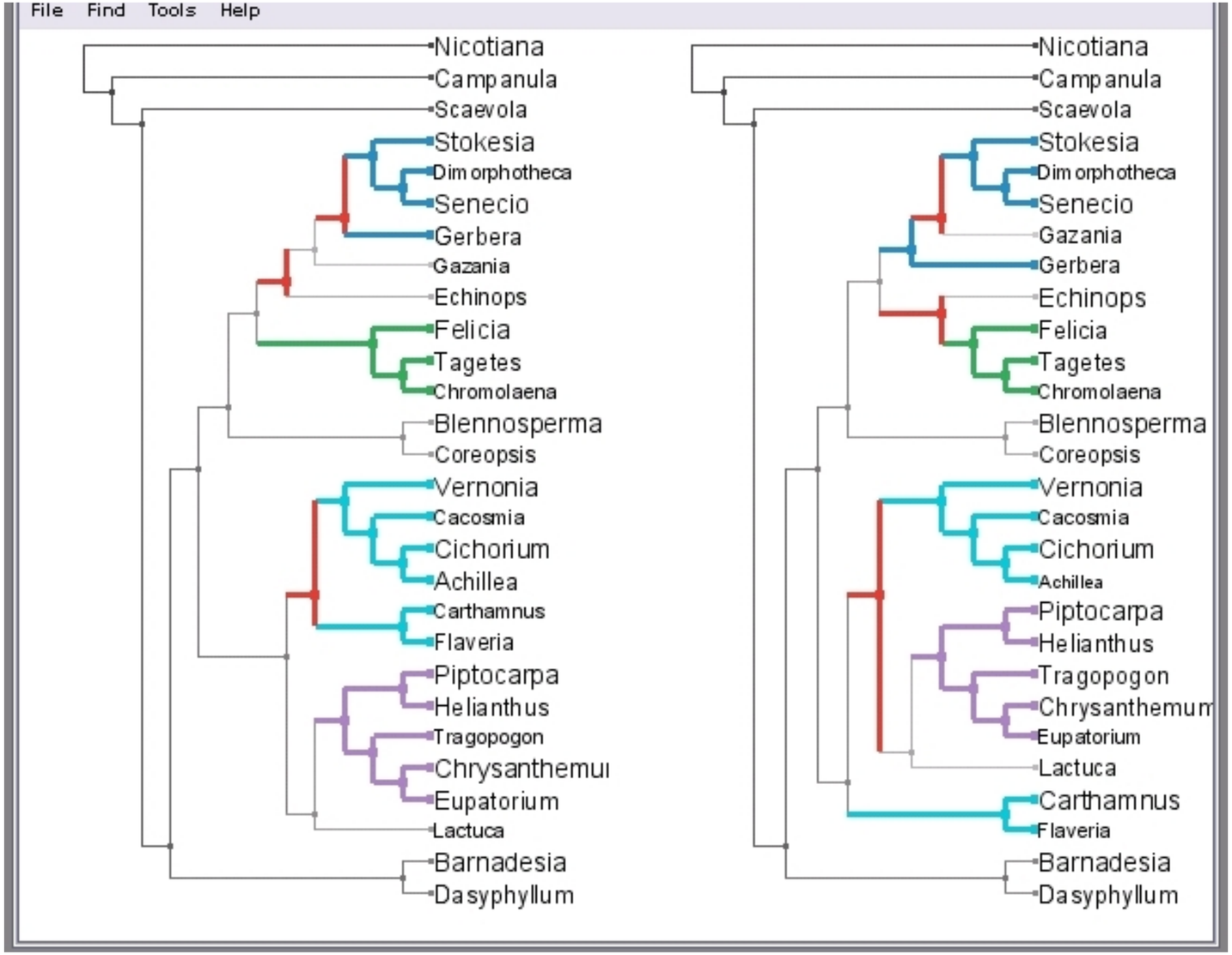


# Collapsible Tree



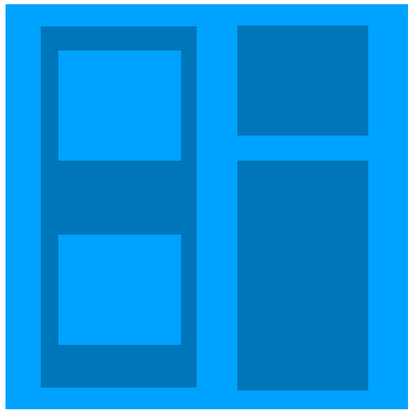

[Heer]

# TreeJuxtaposer




[Munzner et al.]

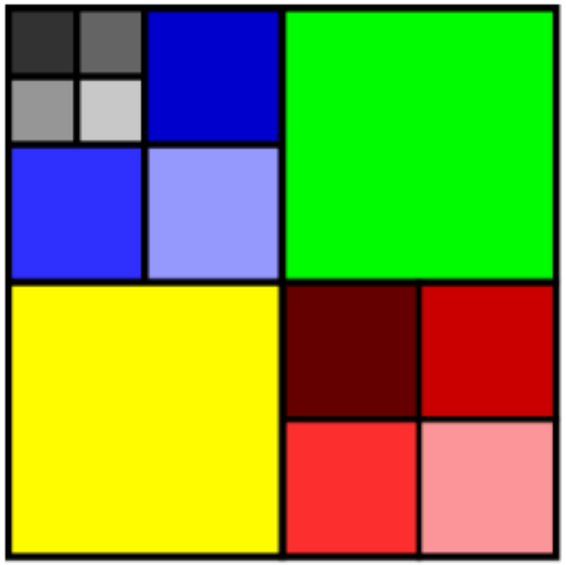
# TreeMap

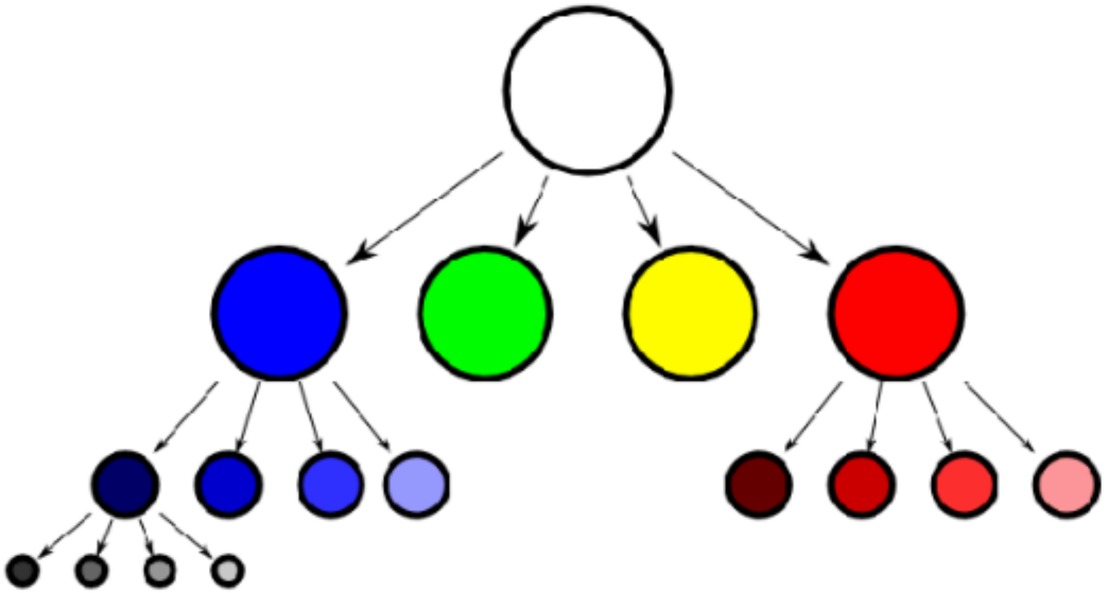


Implicit layout for tree

Display hierarchical (tree-structured) data as a set of nested rectangles

Efficient use of space



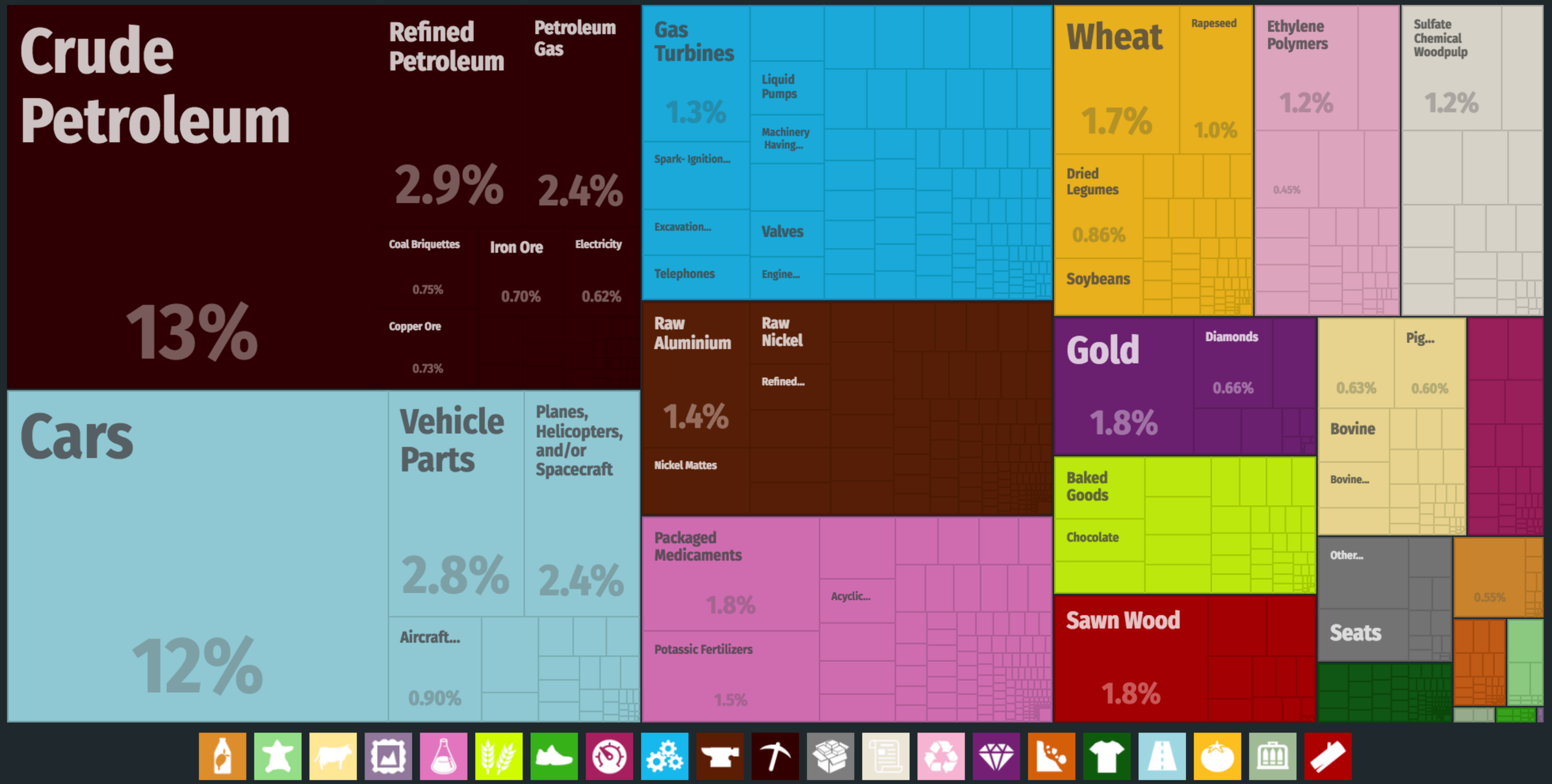


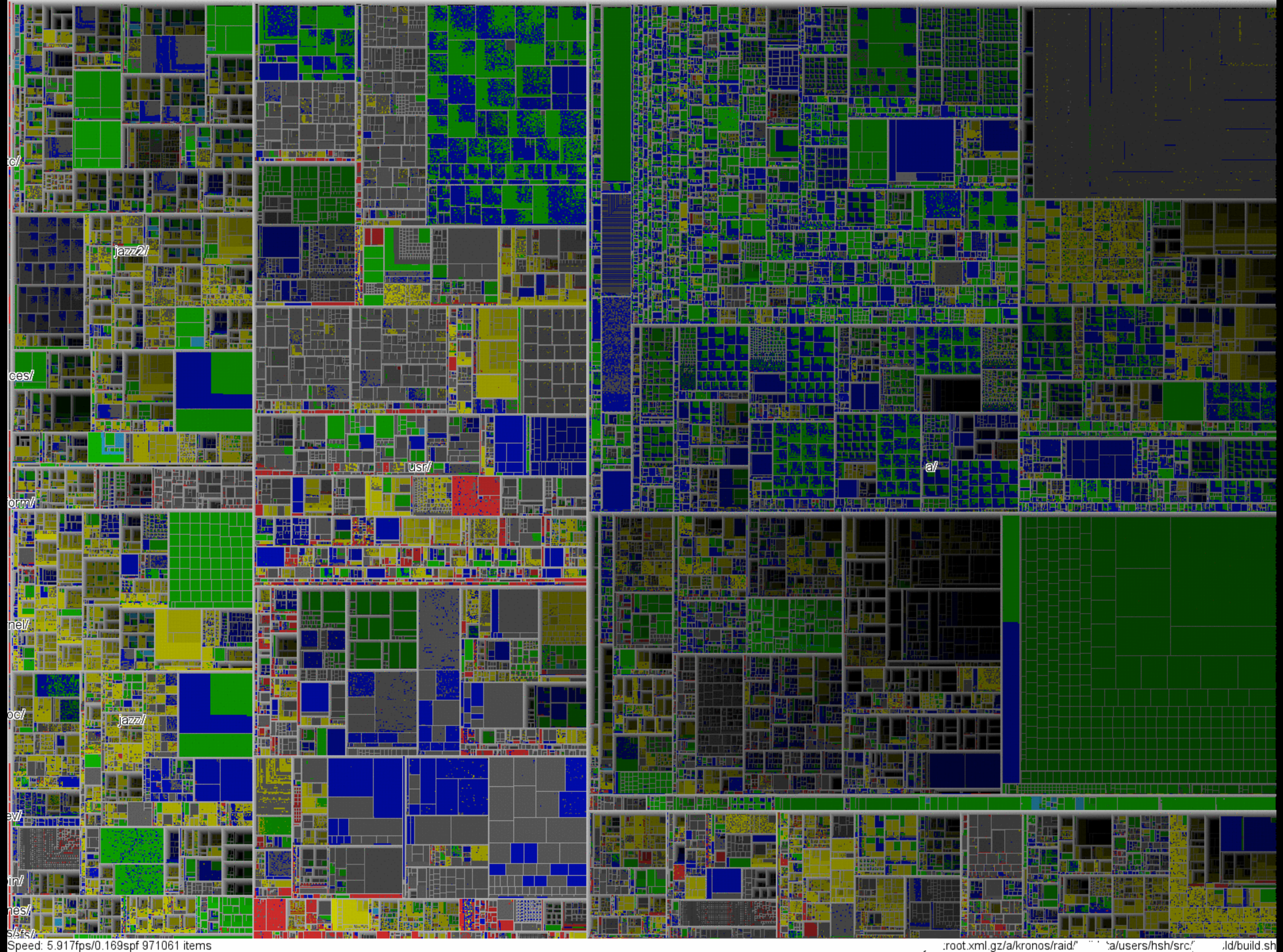
[Johnson and Shneiderman 1991]

< 2014

# What does Canada export? (2015)

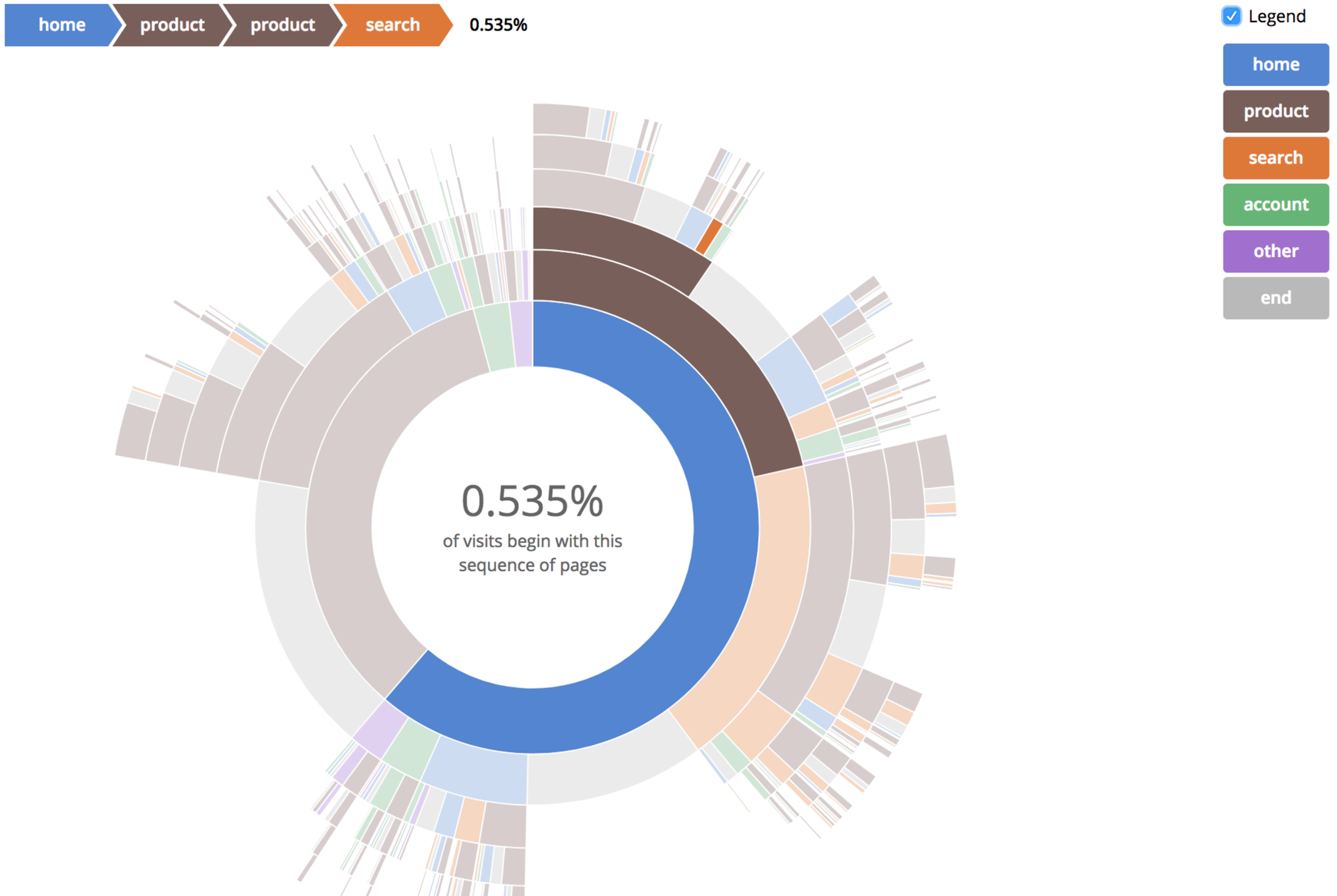
TOTAL: $390B






Speed: 5.917fps/0.169spf 971061 items :root.xml.gz/a/kronos/raid/ a/users/hsh/src/ ld/build.sh

# Radial layout




Viz by Rodden

[Stasko et al.]

# Implicit visualization

**Pros**

Space-efficient because there is no need to draw the edges

Scale well up to large graphs

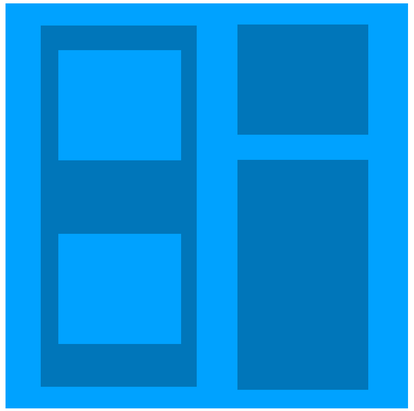
Useful for Attribute based taks on nodes, also useful for simple paths related tasks

**Cons**

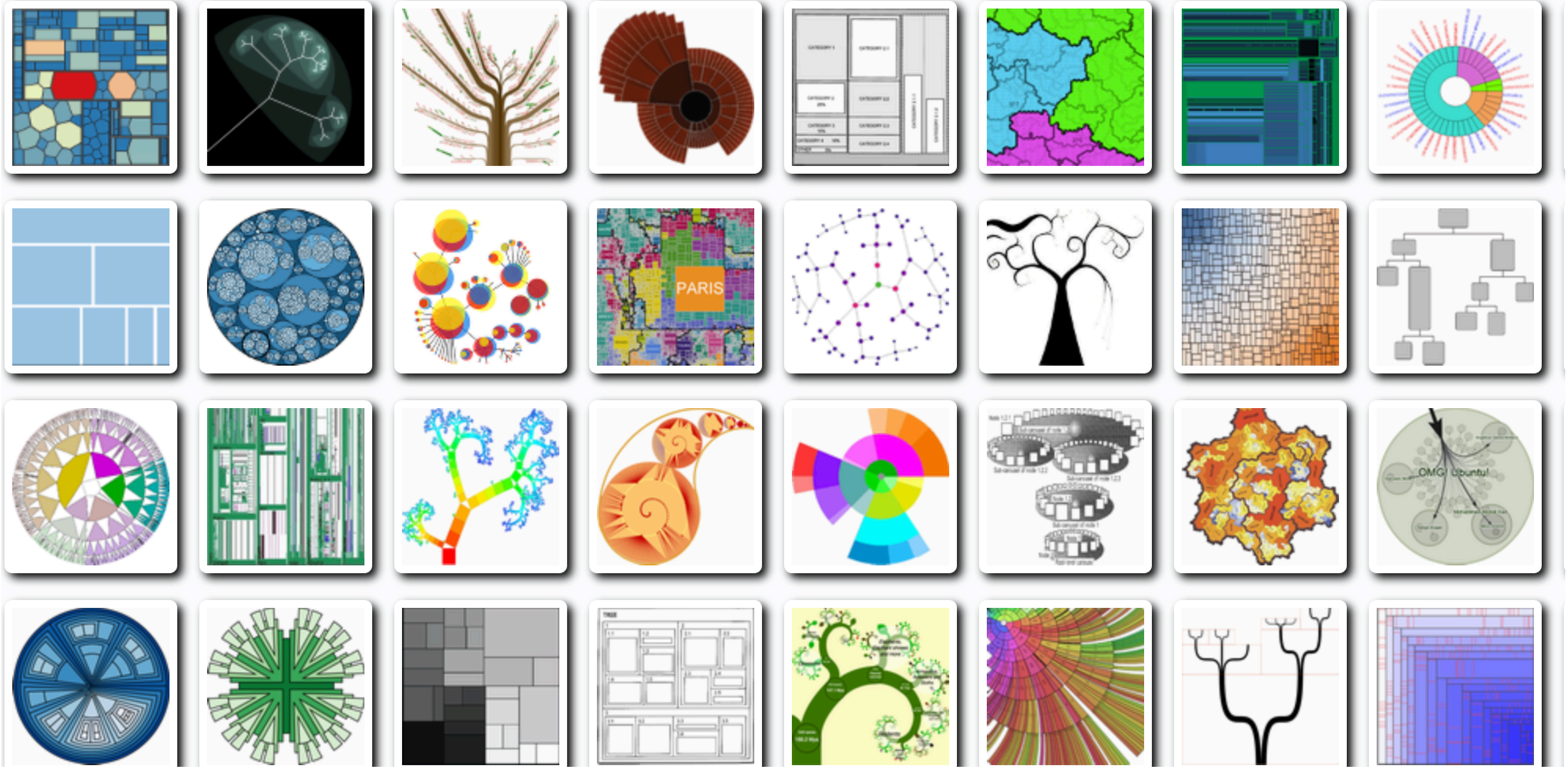
Only works for trees

Node cannot be freely arranged in space, no edges

Spatial relations such as overlap or inclusion lead to occlusion



# More on tree viz



# Tools, libraries

# Cytoscape.js

Graph theory / network library for analysis and visualisation

Supports CommonJS/Node.js/Browserify/Webpack, AMD/Require.js, jQuery, npm, CDNJS, Bower, jspm, Meteor/Atmosphere, and plain JS/JavaScript

GitHub | Download 3.2.3

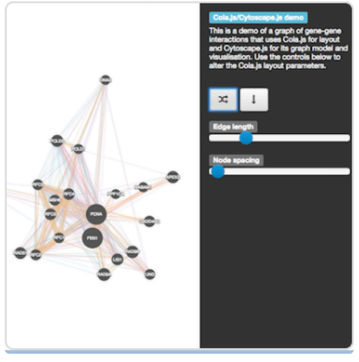
Extensions | News & tutorials | Twitter | Gitter | File a GitHub issue | Contribute | Search previous questions via Stack Overflow (or ask a different question)

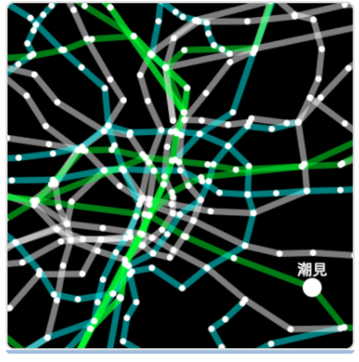
Like Cytoscape.js? Donate | Star it on GitHub | Share it on Twitter | Share it on Google+

Donations are tax deductible to a 501(c)(3) nonprofit organization, The Cytoscape Consoritum, Tax ID: 20-4909879.

Use Cytoscape.js? Let us know

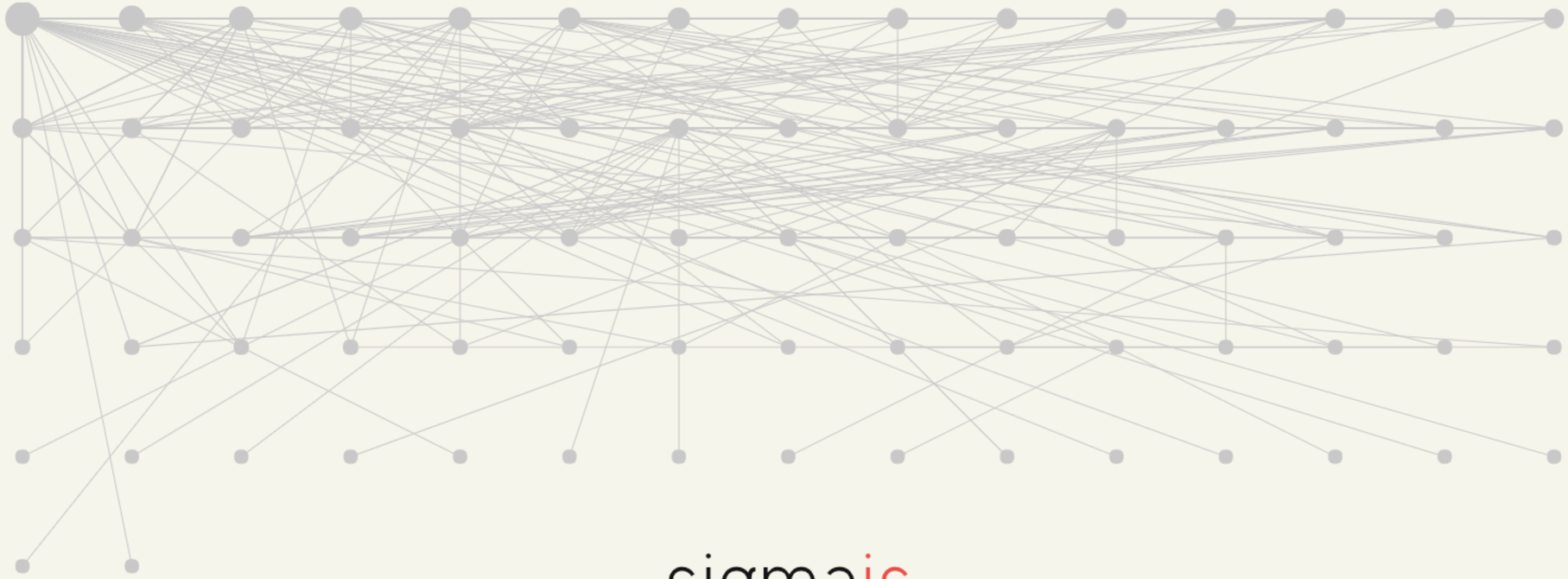
## Demos




sigmajs

GET STARTED FEATURES USE CASES TUTORIAL REFERENCES



sigmajs

TUTORIAL v1.2.0 DOWNLOAD

# The Open Graph Viz Platform

**Gephi is the leading visualization and exploration software for all kinds of graphs and networks. Gephi is open-source and free.**

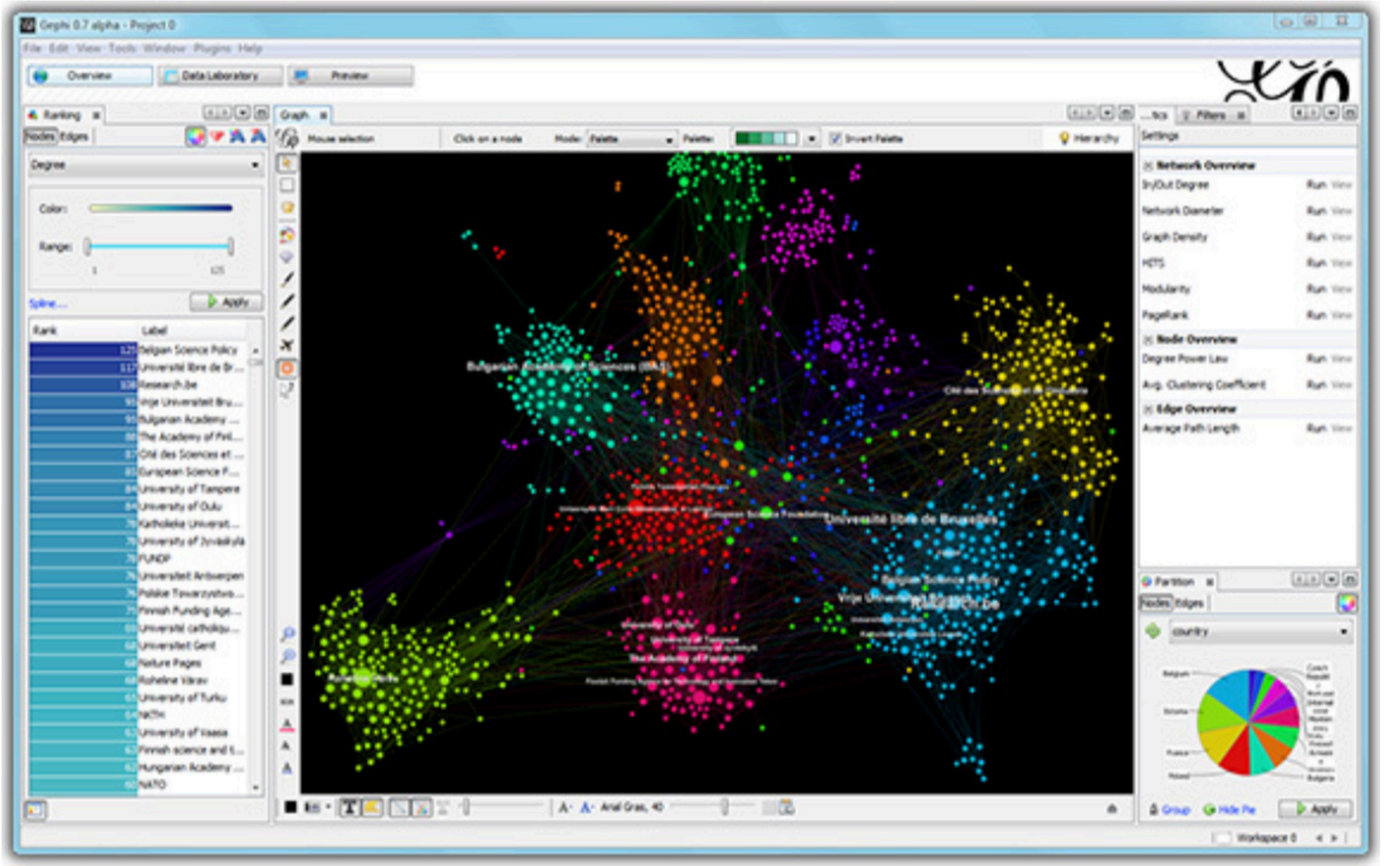
**Runs on Windows, Mac OS X and Linux.**

Learn More on Gephi Platform »

Download FREE
Gephi 0.9.1

Release Notes | System Requirements

- **Features**
- **Quick start**
- **Screenshots**
- **Videos**



# 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 2.22**

See Instructions | See Changelog

## It is *Fast!*

## Extensive Features

## Powerful Visualization

# NetworkX

## Stable (notes)

1.11 — January 2016
download | doc | pdf

## Latest (notes)

2.0 development
github | doc | pdf

## Archive

## Contact

Mailing list
Issue tracker

# Software for complex networks

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

## Features

- 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)
- Edges can hold arbitrary data (e.g., weights, time-series)
- Open source 3-clause BSD license
- Well tested with over 90% code coverage
- Additional benefits from Python include fast prototyping, easy to teach, and multi-platform