

Graphs and Networks 1



CS 7450 - Information Visualization
November 14, 2016
John Stasko

Learning Objectives



- Define network concepts
 - vertex, edge, cycle, degree, direction
- Describe different node-link design choices
 - color, width, position, shape, size, label, form
- Enumerate primary aesthetic considerations for layouts
 - edge crossings, clusters, symmetry, edge lengths
- List example tasks for network data
- Explain "ball of string/hairball" problem
- List common layout approaches and describe characteristics of each
 - hierarchical, force-directed, circular, geo, matrix
- Define "edge bundling"

Connections



- Connections throughout our lives and the world
 - Circle of friends
 - Delta's flight plans
 - ...
- Model connected set as a *Graph*

Fall 2016

CS 7450

3

What is a Graph?



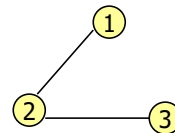
- Vertices (nodes) connected by
- Edges (links)

	1	2	3
1	0	1	0
2	1	0	1
3	0	1	0

Adjacency matrix

Adjacency list

1: 2
2: 1, 3
3: 2



Drawing

Fall 2016

CS 7450

4

Graph Terminology



- Graphs can have *cycles*
- Graph edges can be *directed* or *undirected*
- The *degree* of a vertex is the number of edges connected to it
 - *In-degree* and *out-degree* for directed graphs
- Graph edges can have values (*weights*) on them (nominal, ordinal or quantitative)

Fall 2016

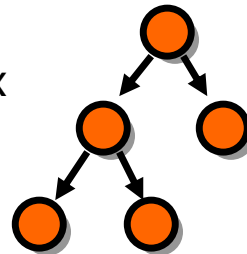
CS 7450

5

Trees are Different



- Subcase of general graph
- No cycles
- Typically directed edges
- Special designated root vertex



Fall 2016

CS 7450

6

Graph Uses



- In information visualization, any number of data sets can be modeled as a graph
 - US telephone system
 - World Wide Web
 - Distribution network for on-line retailer
 - Call graph of a large software system
 - Semantic map in an AI algorithm
 - Set of connected friends
- Graph/network visualization is one of the oldest and most studied areas of InfoVis

Graph Visualization Challenges



- Graph layout and positioning
 - Make a concrete rendering of abstract graph
- Navigation/Interaction
 - How to support user changing focus and moving around the graph

Scale Challenge



- Previous two issues not too bad for small graphs, but large ones are much tougher
- May run out of space for vertices and edges (turns into “ball of string”)
- Can really slow down algorithm

- Sometimes use *clustering* to help
 - Extract highly connected sets of vertices
 - Collapse some vertices together

Fall 2016

CS 7450

9

Navigation/Interaction Challenge



- How do we allow a user to query, visit, or move around a graph?
- Changing focus may entail a different rendering

Fall 2016

CS 7450

10

Layout Examples



- Homework assignment
- Let's judge!

Results

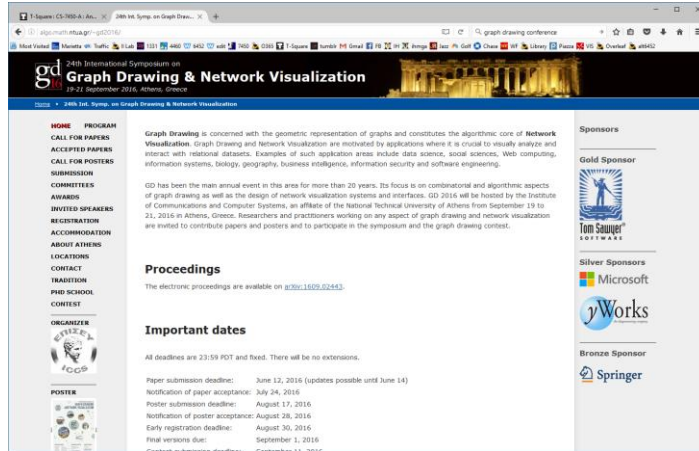


- What led to particular layouts being liked more?
- Discuss

Graph Drawing



Entire research community's focus



Fall 2016

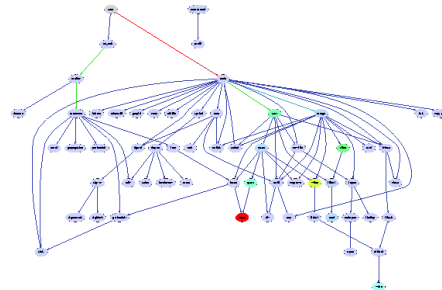
CS 7450

13

Vertex Issues



- Shape
- Color
- Size
- Location
- Label



Fall 2016

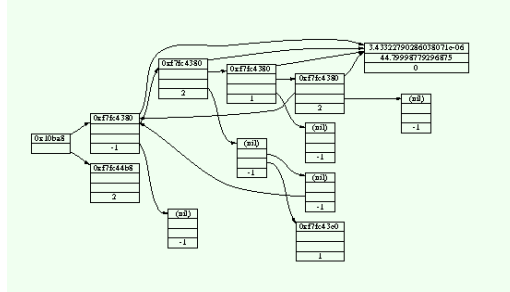
CS 7450

14

Edge Issues



- Color
- Size
- Label
- Form
 - Polyline, straight line, orthogonal, grid, curved, planar, upward/downward, ...



Fall 2016

CS 7450

15

Aesthetic Considerations



- Develop a set of metrics to quantitatively rate the “goodness” of a graph layout
- What metrics would you use?

Fall 2016

CS 7450

16

Aesthetic Considerations



- **Crossings** -- minimize towards planar
- **Total Edge Length** -- minimize towards proper scale
- **Area** -- minimize towards efficiency
- **Maximum Edge Length** -- minimize longest edge
- **Uniform Edge Lengths** -- minimize variances
- **Total Bends** -- minimize orthogonal towards straight-line

Fall 2016

CS 7450

17

Which Matters?



- Various studies examined which of the aesthetic factors matter most and/or what kinds of layout/vis techniques look best
 - Purchase, Graph Drawing '97
 - Ware et al, *Info Vis* 1(2)
 - Ghoniem et al, *Info Vis* 4(2)
 - van Ham & Rogowitz, *TVCG* '08
 - ...
- Results mixed: Edge crossings do seem important

Fall 2016

CS 7450

18

Shneiderman's NetViz Nirvana



- 1) Every node is visible
- 2) For every node you can count its degree
- 3) For every link you can follow it from source to destination
- 4) Clusters and outliers are identifiable

Fall 2016

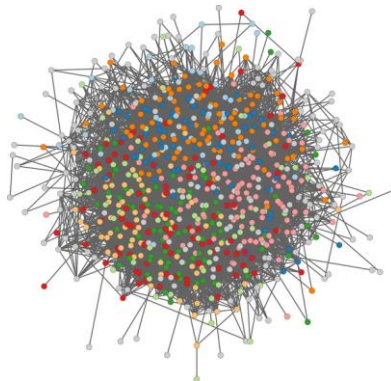
CS 7450

19

Classic Problem



- With enough vertices and enough edges, you get...
- A hairball!
(ball-of-string)



<http://visone.info/wiki/images/b/b7/Caltech36-hairball.png>

Fall 2016

CS 7450

20

But What about User Tasks?



- So what do people want to do with or learn from network visualizations?
 - Recurring theme of this class: Too often this is neglected

Graph Vis Task Taxonomy



- Start with Amar et al '05 low-level tasks (retrieve value, find extreme, sort, etc.)
- Then add four types of other tasks (next pages)

Graph Vis Task Taxonomy



- 1. Topology-based tasks
 - Adjacency
Find the set of nodes adjacent to a node
 - Accessibility
Find the set of nodes accessible to a node
 - Common connection
Given nodes, find the set of nodes connected to all
 - Connectivity
 - Find shortest path
 - Identify clusters
 - Identify connected components

Fall 2016

CS 7450

23

Graph Vis Task Taxonomy



- 2. Attribute-based tasks
 - On the nodes
Find the nodes having a specific attribute value
 - On the edges
Given a node, find the nodes connected only by certain kinds of edges

Fall 2016

CS 7450

24

Graph Vis Task Taxonomy



- 3. Browsing tasks
 - Follow path
Follow a given path
 - Revisit
Return to a previously visited node
- 4. Overview task
 - Compound exploratory task
Estimate size of a network
Find patterns

Fall 2016

CS 7450

25

Graph Drawing Uses



- Many domains and data sets can benefit significantly from nice graph drawings
- Let's look at some examples...

Fall 2016

CS 7450

26

Human Diseases

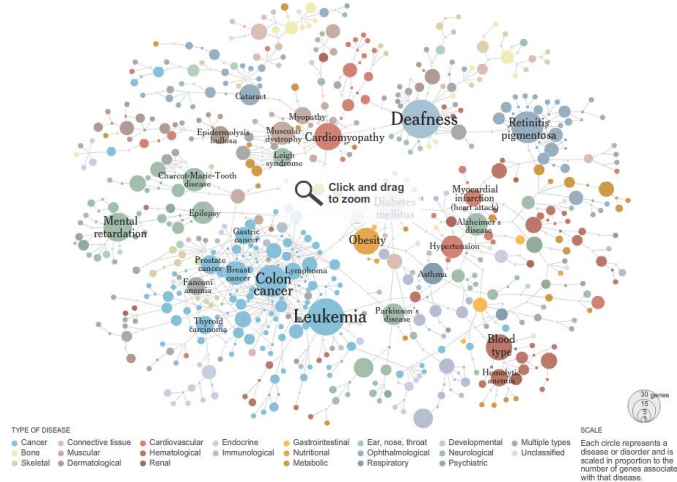
May 5, 2008

SIGN IN TO E-MAIL FEEDBACK



Mapping the Human 'Diseaseome'

Researchers created a map linking different diseases, represented by circles, to the genes they have in common, represented by squares. Related Article: [Redefining Disease, Genes and All](#)



Note the two extra variables per vertex

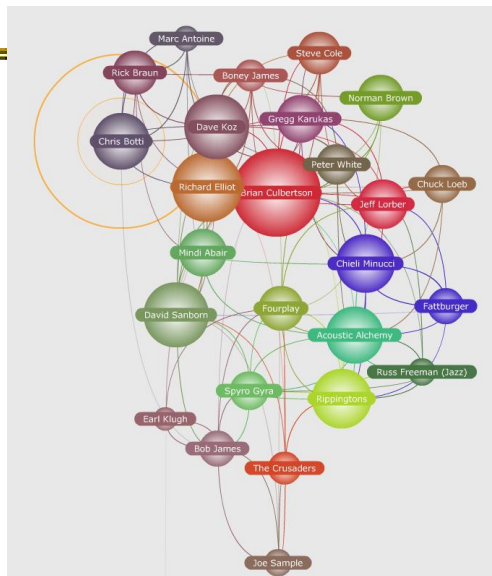
Fall 2016

CS 7450

27

Music Artists

older

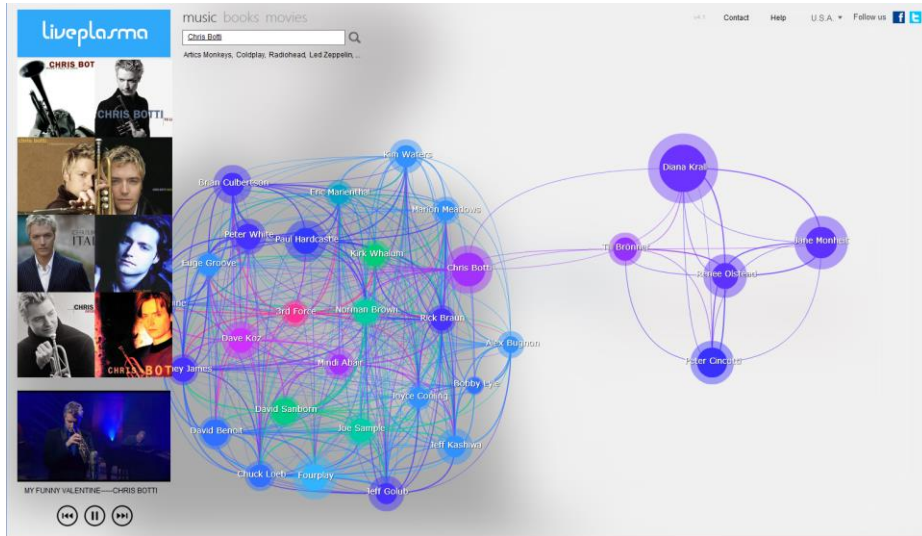


<http://www.liveplasma.com/>

Fall 2016

CS 7450

28



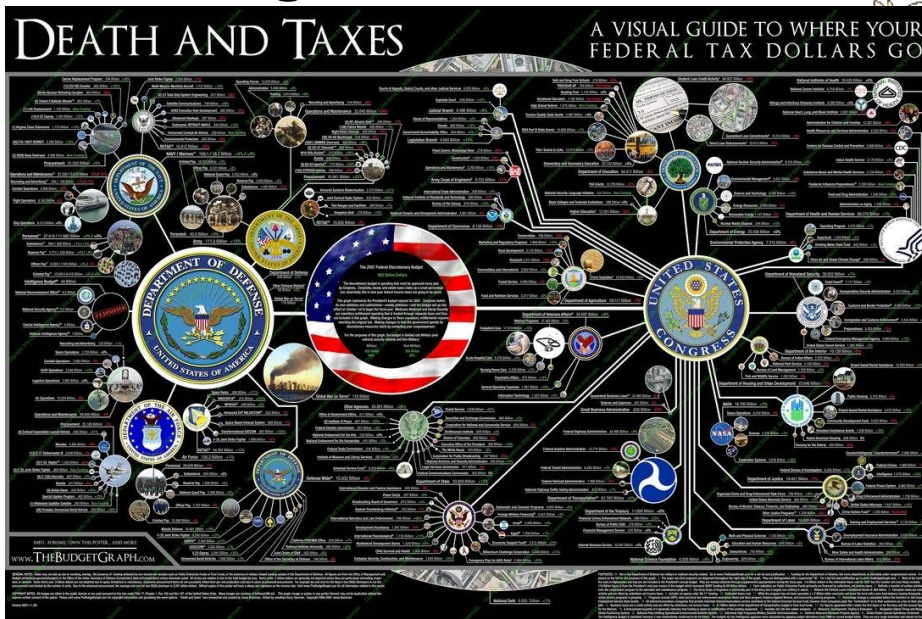
Fall 2016

CS 7450

29

<http://mibi.deviantart.com/art/Death-and-Taxes-2007-39894058>

US Budget



Social Analysis

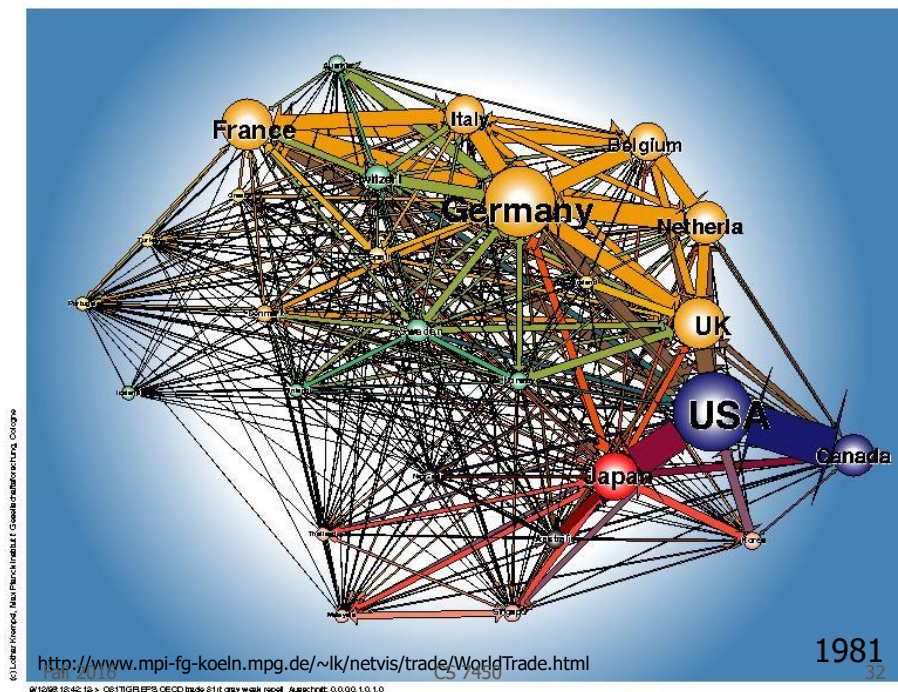


- Facilitate understanding of complex socio-economic patterns
- Social Science visualization gallery (Lothar Krempel):
 - <http://www.mpi-fg-koeln.mpg.de/~lk/netvis.html>
- Next slides: Krempel & Plumper's study of World Trade between OECD countries, 1981 and 1992

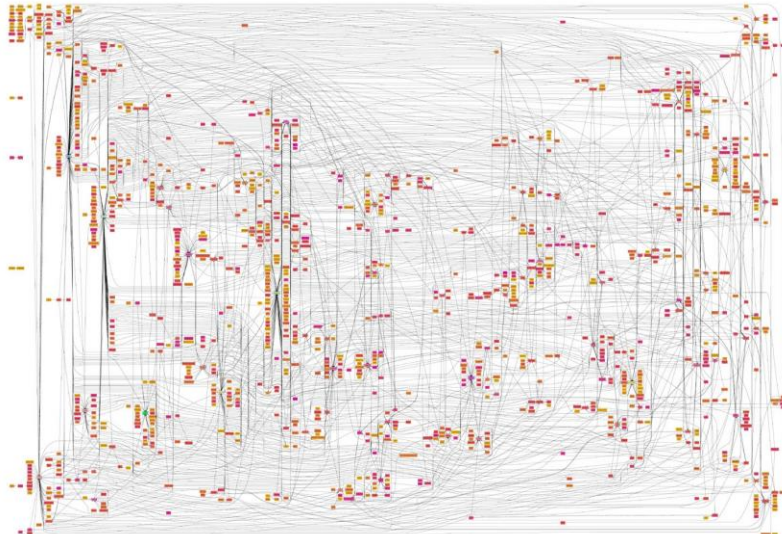
Fall 2016

CS 7450

31



People connections



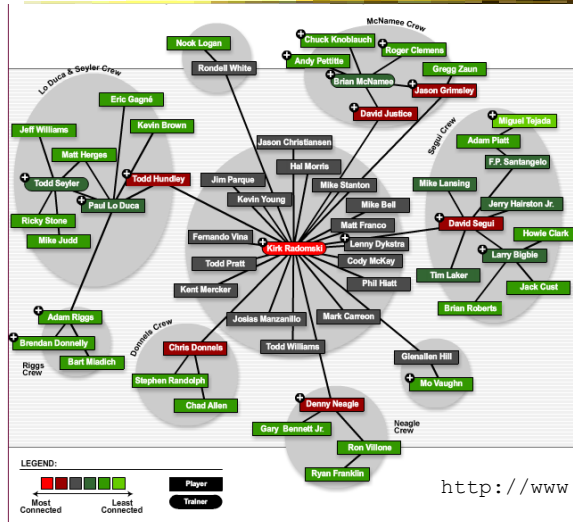
Fall 2016

CS 7450

Charles Isbell, Cobot

35

Steroids in MLB

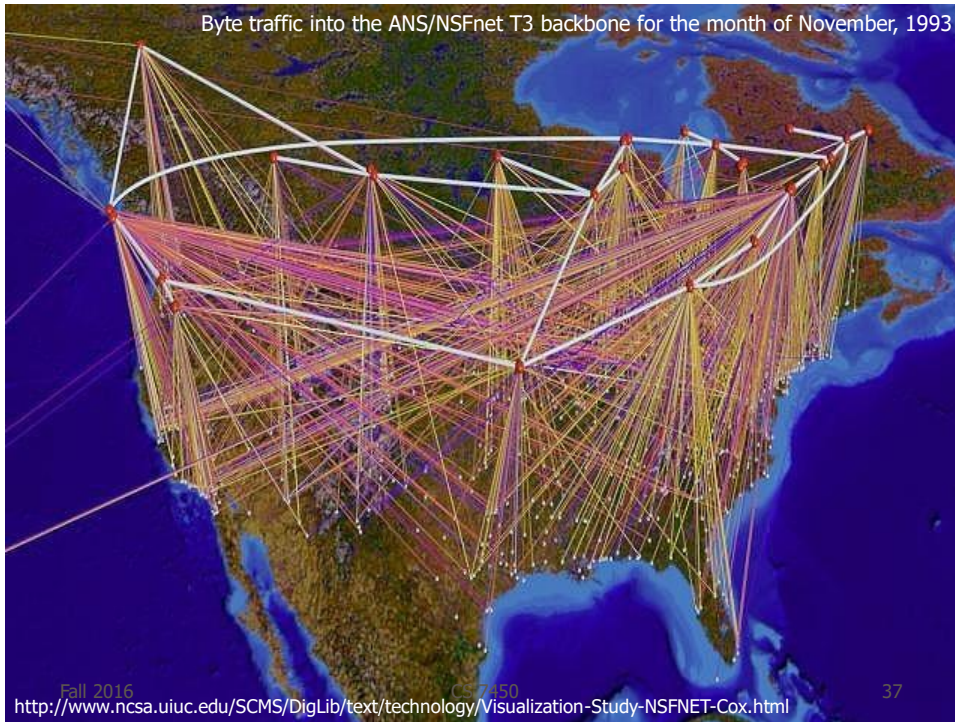


<http://www.slate.com/id/2180392/>

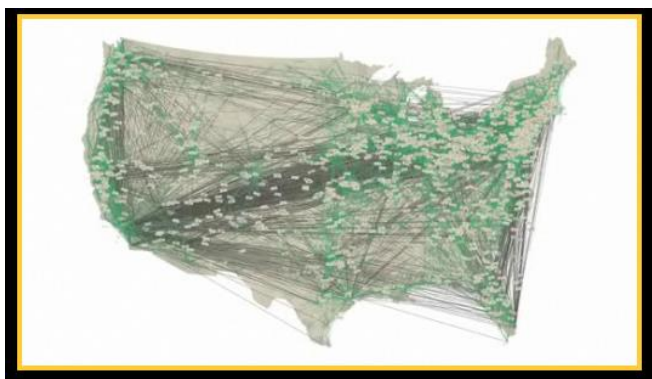
Fall 2016

CS 7450

36



Follow the Money



Where does a dollar bill go?

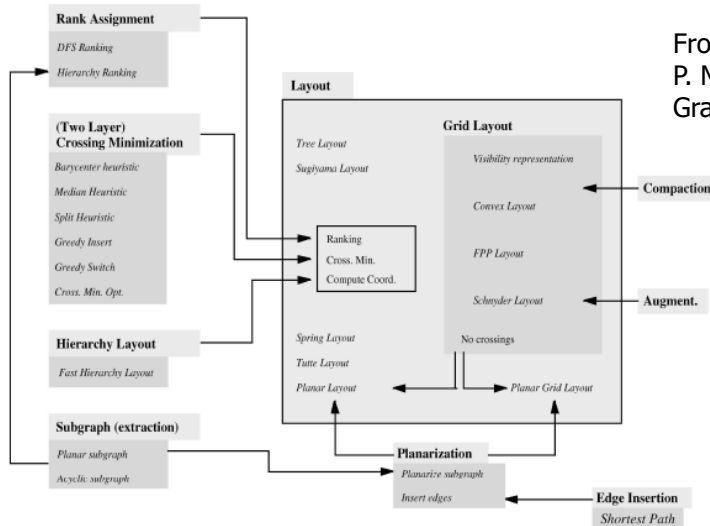
http://www.nsf.gov/news/special_reports/scivis/follow_money.jsp

Layout Heuristics



- Layout algorithms can be
 - polyline edges
 - planar
 - No edge crossings
 - orthogonal
 - horizontal and vertical lines/polylines
 - grid-based
 - vertices, crossings, edge bends have integer coords
 - curved lines
 - hierarchies
 - circular
 - ...

Types of Layout Algorithms



From:
P. Mutzel, et al
Graph Drawing '97

Common Layout Techniques



- Hierarchical
- Force-directed
- Circular
- Geographic-based
- Clustered
- Matrix
- Attribute-based

We will discuss many of these further in the slides to come

Tree Layout (Use Last Week)



- Run a breadth-first search from a vertex
 - This imposes a spanning tree on the graph
- Draw the spanning tree

- Simple and fast, but obviously doesn't represent the whole graph

Hierarchical Layout



Often called Sugiyama layout

Try to impose hierarchy on graph
Reverse edges if needed to
remove cycles
Introduce dummy nodes
Put nodes into layers or levels
Order l->r to minimize crossings

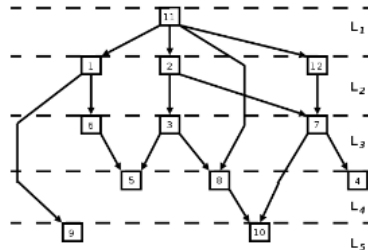


Figure: A graph showing a layered layout, created with the Sugiyama heuristic, with the layers shown. The bends in the edges correspond to dummy nodes.

<http://www.csse.monash.edu.au/hons/se-projects/2006/Kieran.Simpson/output/html/node7.html#sugiyamaexample>

Fall 2016

CS 7450

43

Force-directed Layout



- Example of constraint-based layout technique
- Impose constraints (objectives) on layout
 - Shorten edges
 - Minimize crossings
 - ...
- Define through equations
- Create optimization algorithm that attempts to best satisfy those equations

Fall 2016

CS 7450

44

Force-directed Layout



- Spring model (common)
 - Edges – Springs (gravity attraction)
 - Vertices – Charged particles (repulsion)
- Equations for forces
- Iteratively recalculate to update positions of vertices
- Seeking local minimum of energy
 - Sum of forces on each node is zero

Fall 2016

CS 7450

45

Force-directed Example

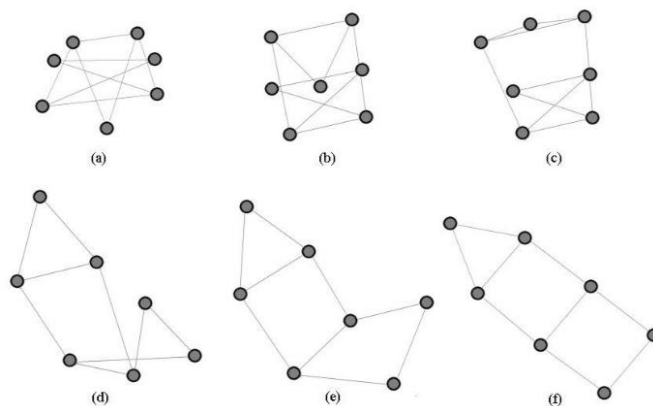


Figure 2. A graph drawing through a number of iterations of a force directed algorithm.

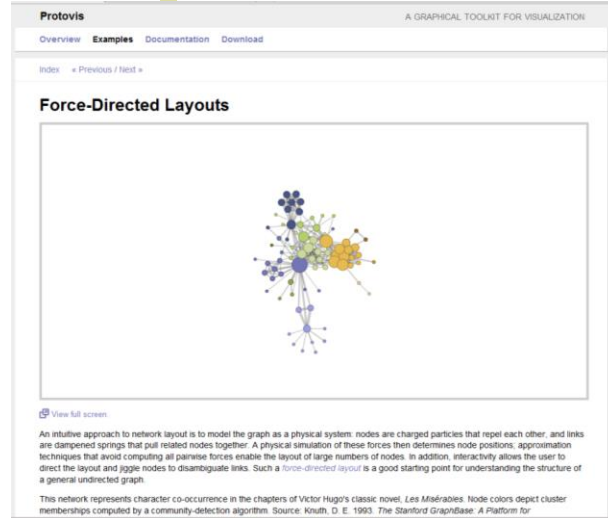
<http://www.cs.usyd.edu.au/~aquigley/3dfade/>

Fall 2016

CS 7450

46

In Action



Fall 2016

CS 7450

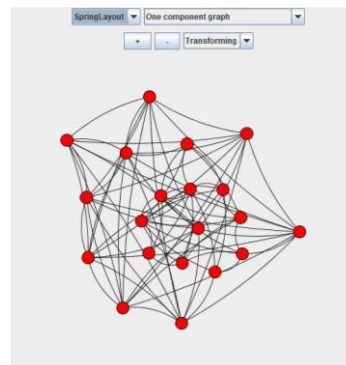
47

Variant

Images from JUNG



- Spring layout
 - Simple force-directed spring embedder



Fall 2016

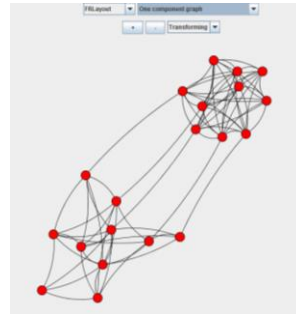
CS 7450

48

Variant



- Fruchterman-Reingold Algorithm
 - Add global temperature
 - If hot, nodes move farther each step
 - If cool, smaller movements
 - Generally cools over time



Fall 2016

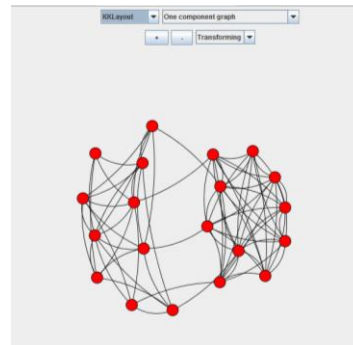
CS 7450

49

Variant



- Kamada-Kawai algorithm
 - Examines derivatives of force equations
 - Brought to zero for minimum energy



Fall 2016

CS 7450

50

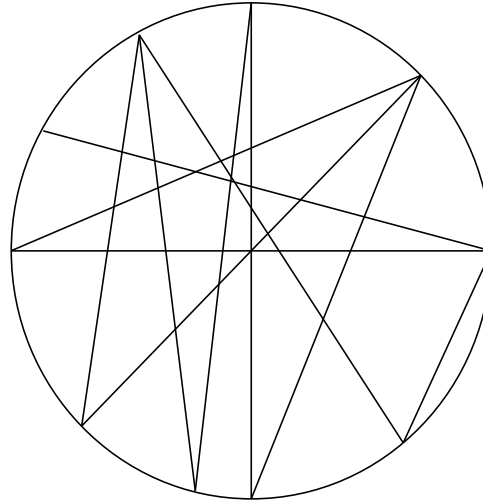
Circular Layout



Ultra-simple
May not look so great

Space vertices out around circle
Draw lines (edges) to connect
vertices

Uses curved lines and
becomes "chord diagrams"



Fall 2016

CS 7450

51

Circos

<http://circos.ca/>

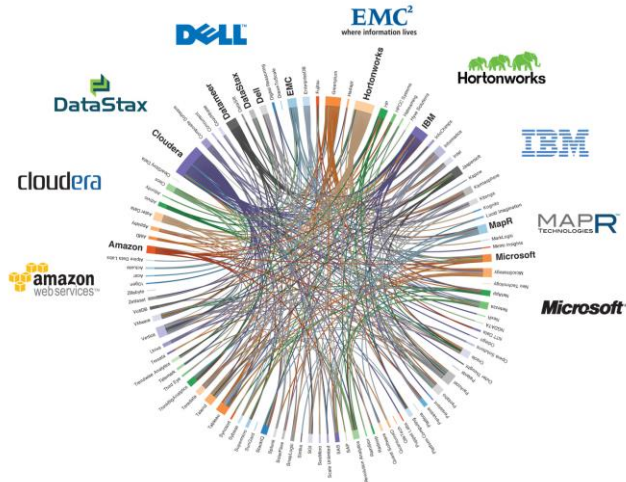


Fall 2016

CS 7450

52

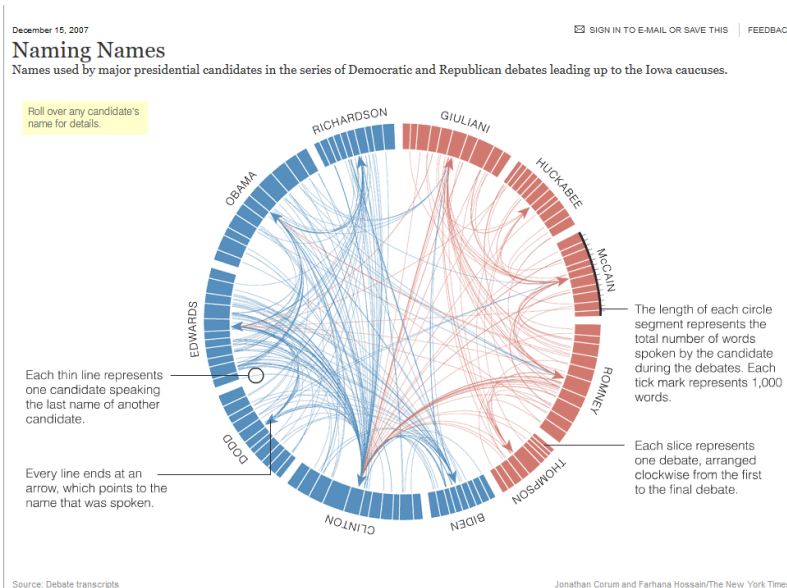
Chord Diagram



Fall 2016

CS 7450

53



Fall 2016

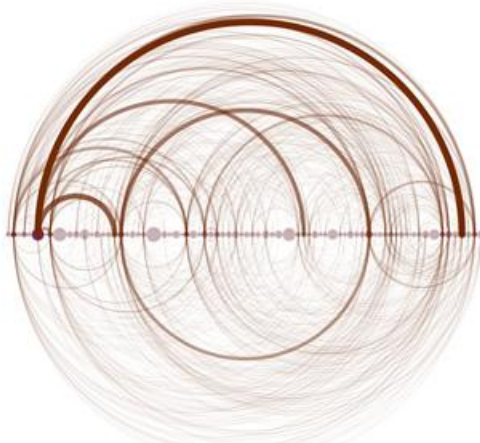
CS 7450

54

Arc Diagram Layout



Wattenberg
InfoVis '02

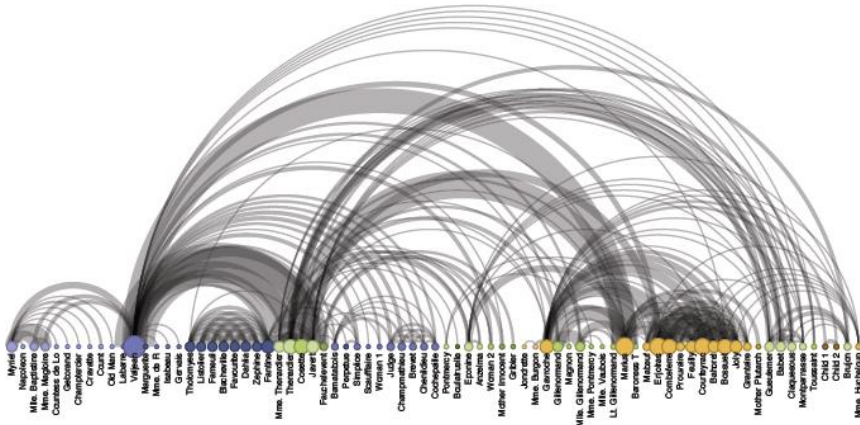


<http://www.visualcomplexity.com/vc/index.cfm?method=Arc%20Diagrams>

Fall 2016

CS 7450

55



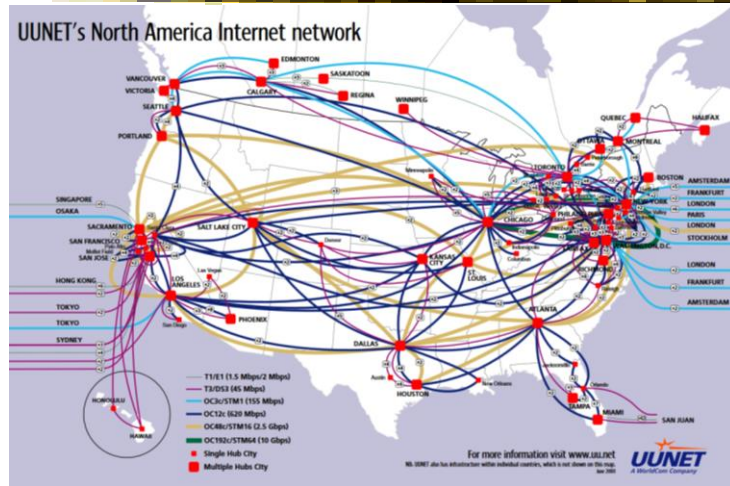
<http://ha.stanford.edu/jheer/files/zoo/ex/networks/arc.html>

Fall 2016

CS 7450

56

Geo/Map Approaches



Maps can easily become networks

Fall 2016

CS 7450

57

Where People Run



CITYLAB | [News](#) | [Columns](#) | [Columns](#) | [Maps](#) | [Photo](#)

COMMUTE WORK HOUSING WEATHER

Mapping Where People Run

These graphics chart the most popular routes in 22 cities.

JENNY XIE | @jxie | Feb 5, 2014 | 33 Comments

Share on Facebook | [Twitter](#) | [LinkedIn](#) | [Google+](#) | [Print](#)

Love CityLab? Make sure you've signed up for our free e-mail newsletter:

Inspired by a 2011 [project](#) that mapped popular running routes in a few European cities, Nathan Yau at [MappingData](#) has done the same for 22 major cities, including 18 in the U.S.

To make these maps, Yau simply grabbed public data from the exercise-tracking app [Strava](#). While these visualizations are not representative of all runners in a city, they do offer useful information on urban spaces. For one, we see that people really do love running near water and in parks.

Here's a selection of Yau's work. See the rest [here](#).

Boston

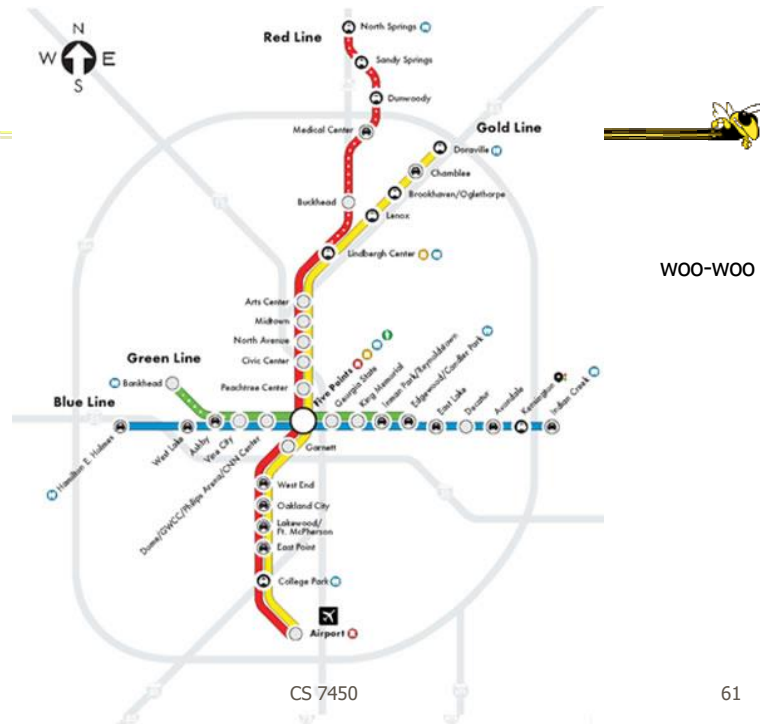
Atlanta



Fall 2016

CS 7450

58



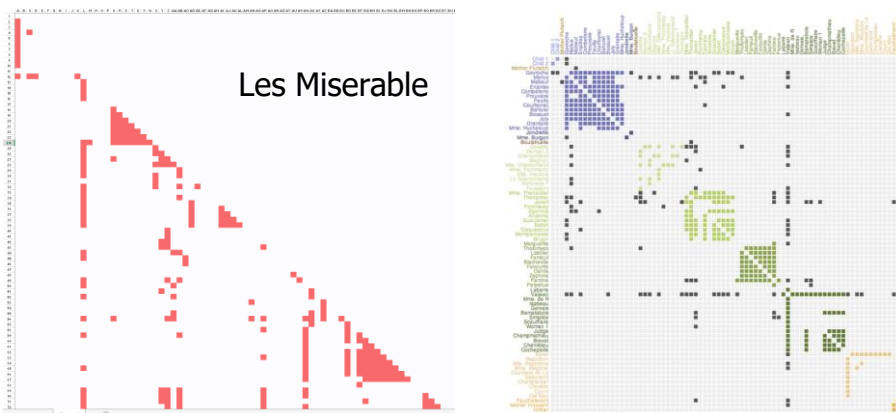
3 Subway Diagrams

- Geographic landmarks largely suppressed on maps, except water (rivers in London & Paris) and asphalt (highways in Atlanta)
 - Rather fitting, no?
- These are more *graphs* than maps!
- Subway-style diagrams have become their own genre of network layouts

Matrix Representations



- Forget the node-link approach



<http://www.themacroscope.org>

<https://homes.cs.washington.edu/~jheer/files/zoo/>

Fall 2016

CS 7450

63

Drawing Graphs Better



- Can we do clever “tricks” to make dense graphs more readable?

Fall 2016

CS 7450

64

Hierarchical Edge Bundles



- Bundle edges that go from/to similar nodes together
 - Like wires in a house
- Uses B-spline curves for edges
- Reduces the clutter from many edges

Holten
TVCG (InfoVis) '06

Fall 2016

CS 7450

65

Example

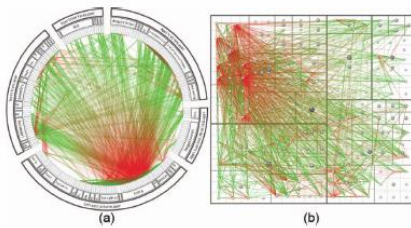


Fig. 11. A software system and its associated call graph (caller = green, callee = red). (a) and (b) show the system without bundling using a radial and a squarified treemap layout (node labels disabled), respectively. (a) and (b) mainly show hot spots; the actual connectivity information is more difficult to discern due to visual clutter.

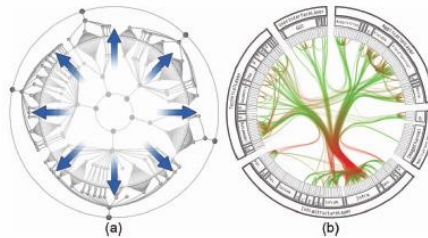


Fig. 12. Radial layout construction. (a) A radial tree layout is used for the inner circle and subsequently mirrored to the outside; (b) the inner layout is hidden and its structure is used to guide the adjacency edges. An icicle plot based on the mirrored layout is used to show the hierarchy.

Fall 2016

CS 7450

66

Example

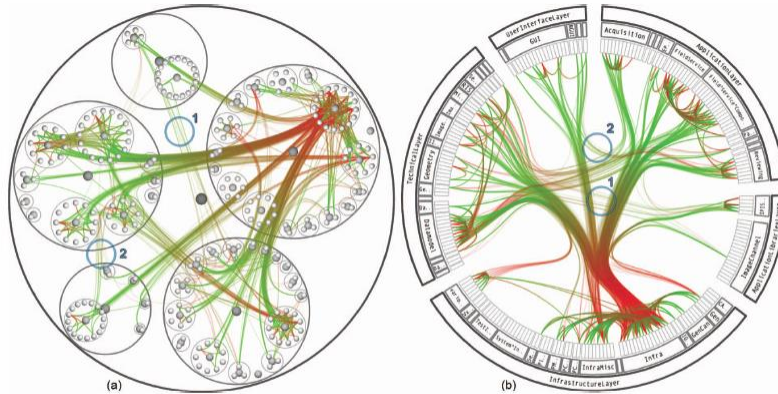


Fig. 13. A software system and its associated call graph (caller = green, callee = red). (a) and (b) show the system with bundling strength $\beta = 0.85$ using a balloon layout (node labels disabled) and a radial layout, respectively. Bundling reduces visual clutter, making it easier to perceive the actual connections than when compared to the non-bundled versions (figures 2a and 11a). Bundled visualizations also show relations between sparsely connected systems more clearly (encircled regions); these are almost completely obscured in the non-bundled versions. The encircled regions highlight identical parts of the system for (a), (b), and figure 15.

Fall 2016

CS 7450

67

Example Design Challenge



- Email
- How would you visualize all email traffic in CoC between pairs of people?
- Solutions???

Fall 2016

CS 7450

68

Possible Solutions



- Put everyone on circle, lines between
 - Color or thicken line to indicate magnitude
- Use spring/tension model
 - People who send a lot to each other are drawn close together
 - Shows clusters of communications

Fall 2016

CS 7450

69

Opinion



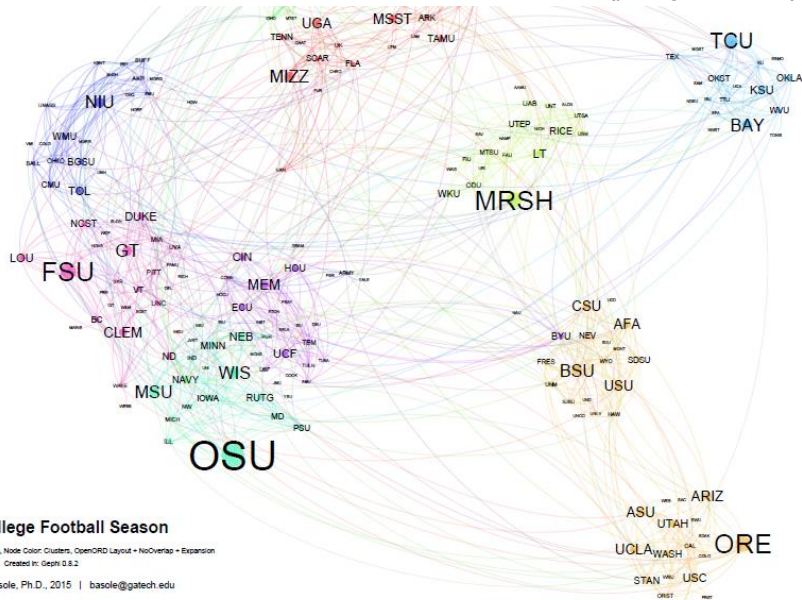
- Many graph drawings/visualizations (particularly the hairballs) provide little insight about the underlying data
 - Many are just "show offs" to make an accompanying visualization

Fall 2016

CS 7450

70

(picking on a friend)

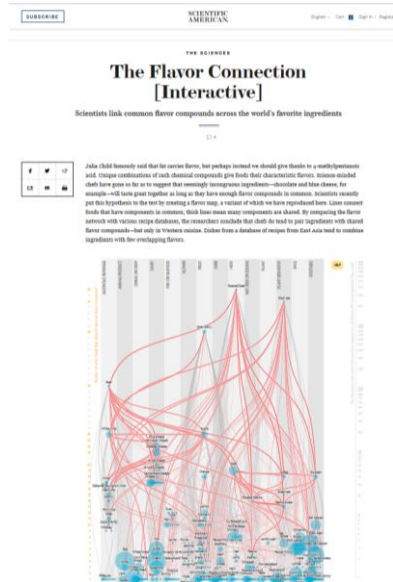


2014 College Football Season
 Node Size: # of Wins, Node Color: Clusters, OpenCRO Layout + NoVertex + Expansion
 Data Source: ESPN Created In: Gephi 0.8.2
 (c) Rahul C. Basole, Ph.D., 2015 | basole@gatech.edu

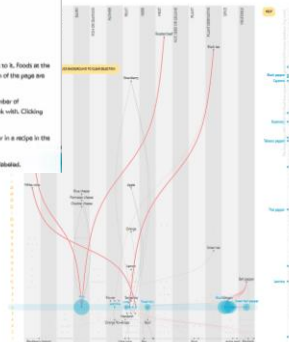
Fall 2016

CS 7450

71



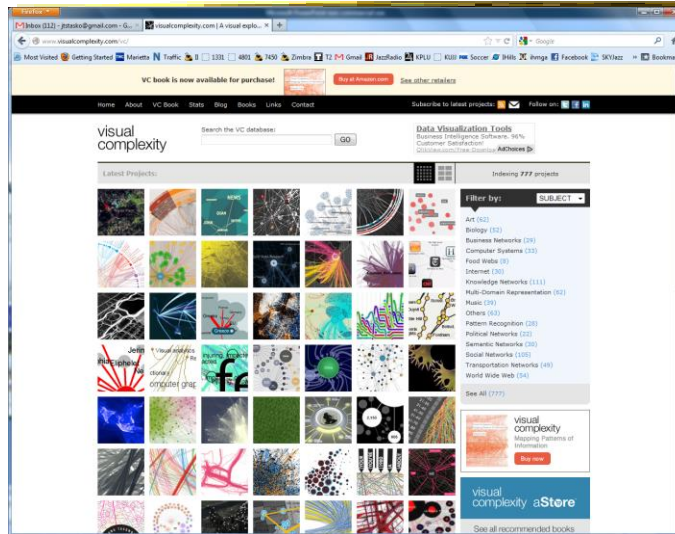
But interaction can help



<http://www.scientificamerican.com/article/flavor-connection-taste-map-interactive/>
 Fall 2016 CS 7450

72

Many Examples



Fall 2016

CS 7450

73

But Is It InfoVis?

- I generally don't consider a pure graph layout (drawing) algorithm to be InfoVis
 - Nothing wrong with that, just an issue of focus
- For InfoVis, I like to see some kind of interaction or a system or an application...
 - Still, understanding the layout algorithms is very important for infovis
 - Let's look at a few...

Fall 2016

CS 7450

74

Graph Drawing Support



- Libraries
 - JUNG (Java Universal Network/Graph Framework)
 - Graphviz (formerly dot?)
- Systems
 - Gephi
 - TouchGraph

Fall 2016

CS 7450

75

<http://jung.sourceforge.net/>

JUNG



The screenshot shows the JUNG project page on SourceForge. The page title is "JUNG - Java Universal Network/Graph Framework". The main content area is titled "Overview" and contains the following text:

Overview

JUNG - the Java Universal Network/Graph Framework - is a software library that provides a common and extendible language for the modeling, analysis, and visualization of data that can be represented as a graph or network. It is written in Java, which allows JUNG-based applications to make use of the extensive built-in capabilities of the Java API, as well as those of other existing third-party Java libraries.

The JUNG architecture is designed to support a variety of representations of entities and their relations, such as directed and undirected graphs, multi-modal graphs, graphs with parallel edges, and hypergraphs. It provides a mechanism for annotating graphs, entities, and relations with metadata. This facilitates the creation of analytic tools for complex data sets that can examine the relations between entities as well as the metadata attached to each entity and relation.

The current distribution of JUNG includes implementations of a number of algorithms from graph theory, data mining, and social network analysis, such as routines for clustering, decomposition, optimization, random graph generation, statistical analysis, and calculation of network distances, flows, and importance measures (centrality, PageRank, HITS, etc.).

JUNG also provides a visualization framework that makes it easy to construct tools for the interactive exploration of network data. Users can use one of the layout algorithms provided, or use the framework to create their own custom layouts. In addition, filtering mechanisms are provided which allow users to focus their attention, or their algorithms, on specific portions of the graph.

As an open-source library, JUNG provides a common framework for graph/network analysis and visualization. We hope that JUNG will make it easier for those who work with relational data to make use of one another's development efforts, and thus avoid continually re-inventing the wheel.

— The JUNG Framework Development Team

Announcements

- 24 January 2010: [JUNG 2.0.1](#) released.
- 10 April 2009: [JUNG 2.0](#) released.
- 25 July 2008: [JUNG 2.0 beta1](#) released.
- 20 February 2007: [JUNG 2.0 alpha1](#) released.

Jung 2.0 Demos available as applets.
Maven2 generated site for Jung 2.0.

- 28 January 2007: [JUNG 1.7.6](#) released (includes a number of bug fixes).
- 20 October 2006: [JUNG 1.7.5](#) released.

NOTE: This is expected to be the last release of JUNG before JUNG 2.0 (currently in development) is released. We will continue to support JUNG 1.7.5 for the foreseeable future, but future development will be focused solely on JUNG 2.0 and its successors.

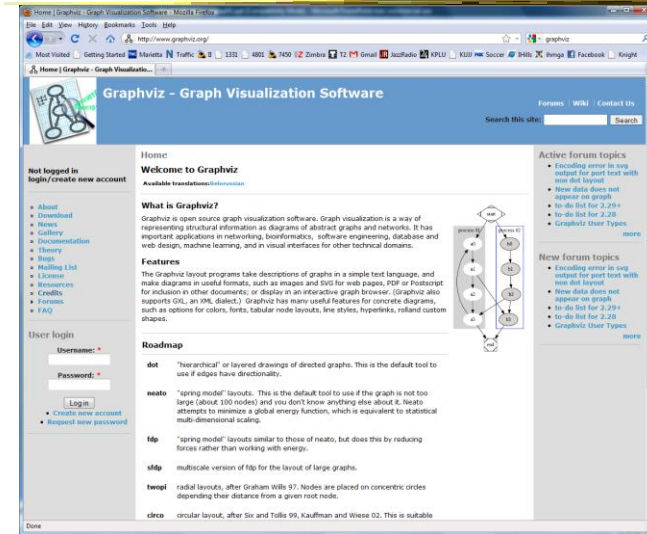
- 7 March 2006: [JUNG 1.7.4](#) released.

Fall 2016

CS 7450

76

Graphviz

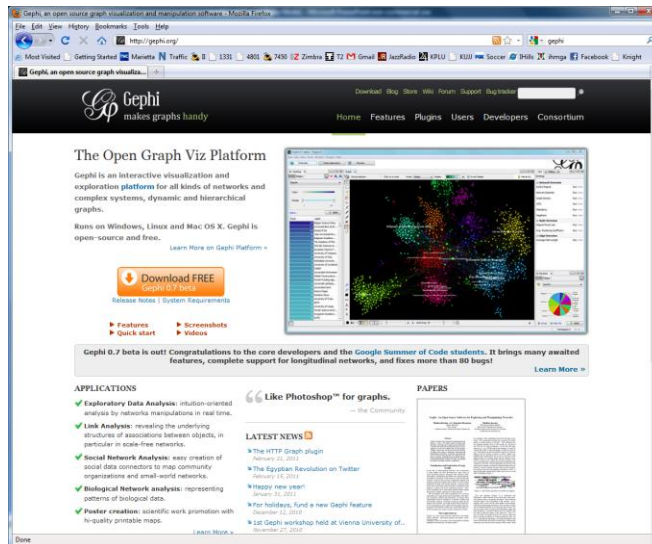


Fall 2016

CS 7450

77

Gephi

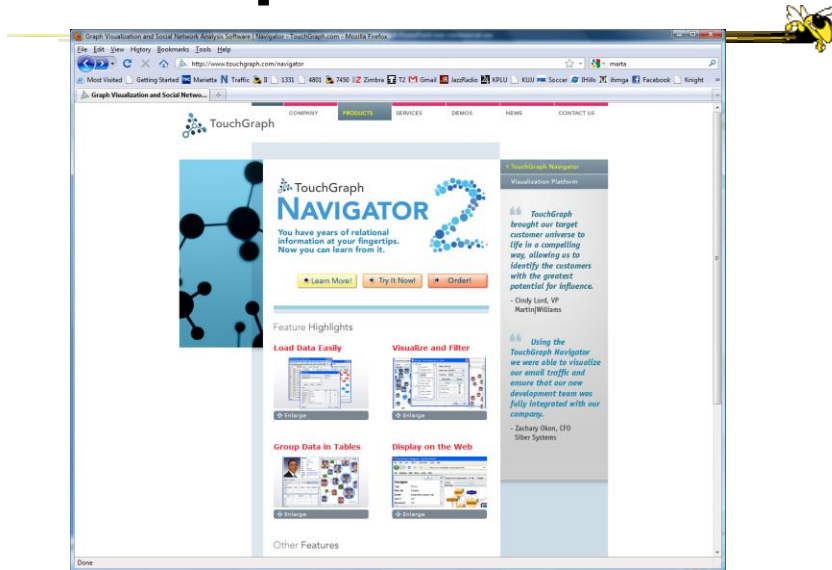


Fall 2016

CS 7450

78

TouchGraph

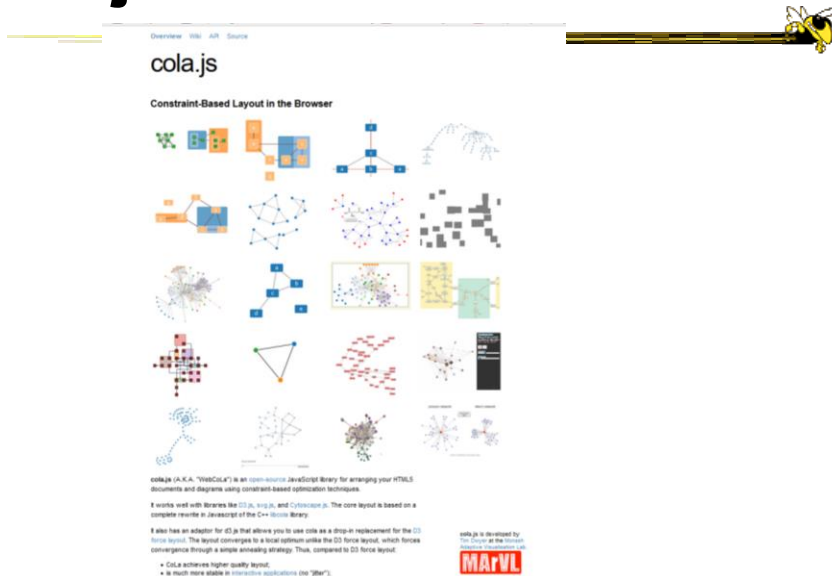


Fall 2016

CS 7450

79

cola.js



Fall 2016

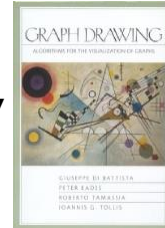
CS 7450

80

Graph Drawing Resources



- Book
 - diBattista, Eades, Tamassia, and Tollis, *Graph Drawing: Algorithms for the Visualization of Graphs*, Prentice Hall, 1999
- Tutorial (talk slides)
 - <http://www.cs.brown.edu/people/rt/papers/gd-tutorial/gd-constraints.pdf>
- Web links
 - <http://graphdrawing.org>



Fall 2016

CS 7450

81

Learning Objectives



- Define network concepts
 - vertex, edge, cycle, degree, direction
- Describe different node-link design choices
 - color, width, position, shape, size, label, form
- Enumerate primary aesthetic considerations for layouts
 - edge crossings, clusters, symmetry, edge lengths
- List example tasks for network data
- Explain "ball of string/hairball" problem
- List common layout approaches and describe characteristics of each
 - hierarchical, force-directed, circular, geo, matrix
- Define "edge bundling"

Fall 2016

CS 7450

82

Reading



- Meirelles chapter 2

Fall 2016

CS 7450

83

Upcoming



- Graphs and Networks 2
- Time Series Data

Fall 2016

CS 7450

84