CS 519: Scientific Visualization

Network Visualization

Eric Shaffer

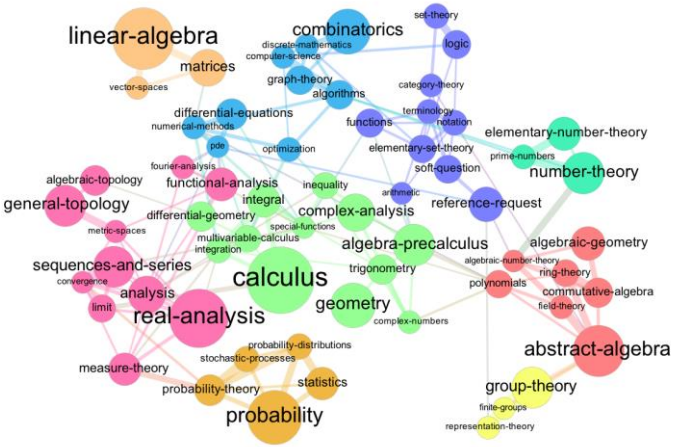
Some slides adapted from:
Alexandru Telea, *Data Visualization Principles and Practice*
Professor Tamara Munzner, University of British Columbia

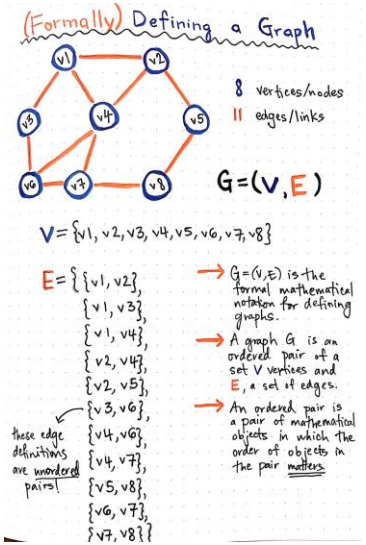
Network Visualization

Graph drawing is an area of mathematics and computer science combining methods from geometric graph theory and information visualization to derive two-dimensional depictions of graphs arising from applications such as social network analysis, cartography, linguistics, and bioinformatics.[1]

A drawing of a graph or **network diagram** is a pictorial representation of the vertices and edges of a graph.

- Wikipedia

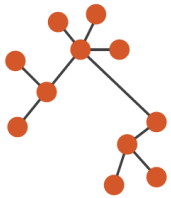




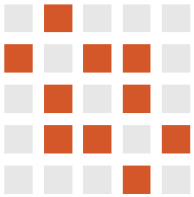
Network Visualization

Arrange networks and trees

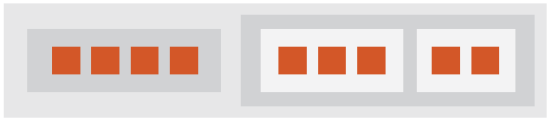
- **Node–Link Diagrams**
  Connection Marks
  ✔ NETWORKS ✔ TREES



- **Adjacency Matrix**
  Derived Table
  ✔ NETWORKS ✔ TREES



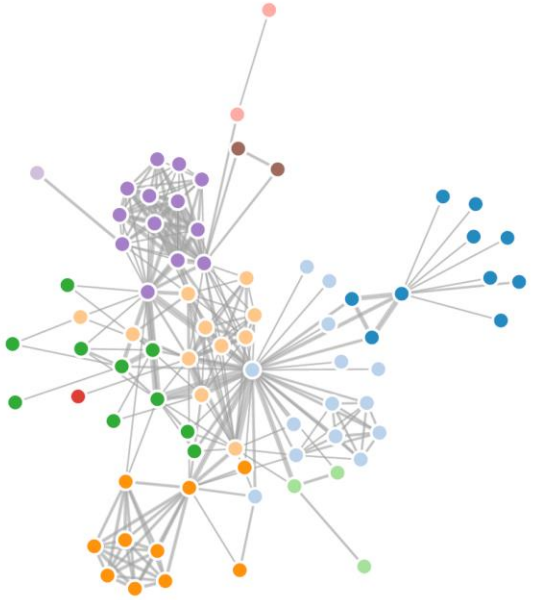
- **Enclosure**
  Containment Marks
  ✘ NETWORKS ✔ TREES



Force-Directed Graph Layout

Idiom: **force-directed placement**

- visual encoding
  - link connection marks, node point marks
- considerations
  - spatial position: no meaning directly encoded
    - left free to minimize crossings
  - proximity semantics?
    - sometimes meaningful
    - sometimes arbitrary, artifact of layout algorithm
    - tension with length
      - long edges more visually salient than short
- tasks
  - explore topology; locate paths, clusters
- scalability
  - node/edge density E < 4N



http://mbostock.github.com/d3/ex/force.html

Force-Directed Layouts

- Can be applied to general graphs (not just directed or trees, etc.)

- design an energy function $E : \mathbf{R}^m \rightarrow \mathbf{R}_+$ which is low when layout is 'good'
  - connected nodes should be close
  - layout distance should reflect graph-theoretic distance
  - nodes should not overlap
  - aspect ratio should be balanced

- define E in terms of forces

[Fruchterman and Reingold '91]

$$\mathbf{F}_a(n_i, n_j) = \frac{|\mathbf{p}_i - \mathbf{p}_j|^2}{k},$$
$$\mathbf{F}_r(n_i, n_j) = -\frac{k^2}{|\mathbf{p}_i - \mathbf{p}_j|}$$

[Eades '84]

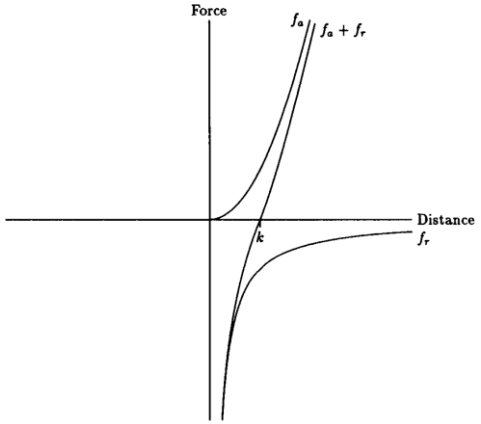
$$\mathbf{F}_a(n_i, n_j) = k \log |\mathbf{p}_i - \mathbf{p}_j|$$
$$\mathbf{F}_r(n_i, n_j) = -\frac{k}{|\mathbf{p}_i - \mathbf{p}_j|^2}.$$

- find node positions p_i by minimizing E:
  - move nodes iteratively along F with some small distance
  - compute F_r only w.r.t. close nodes (to save time)
  - use spatial search structures (e.g. octrees) to find close nodes
  - stop when E low enough or max #iterations reached

Fruchterman-Reingold

$$\mathbf{F}_a(n_i, n_j) = \frac{|\mathbf{p}_i - \mathbf{p}_j|^2}{k},$$
$$\mathbf{F}_r(n_i, n_j) = -\frac{k^2}{|\mathbf{p}_i - \mathbf{p}_j|}$$

- We move n_i according to the force vectors above
  - Move along the line between n_i and n_j
- F_a (attraction) only applies to n_i and n_j that share an edge
- F_r (repulsion) applies to all pairs of vertices
- K is a constant, original paper uses $k = c\sqrt{\frac{WL}{|V|}}$
  - W=width of canvas
  - L=length of canvas
  - |V| is number of vertices
  - C is "found experimentally"
  - Goal is to distribute vertices uniformly
    - But with connected vertices close to each other
  - K is radius of empty circle around a vertex…
  - K is the distance at which the forces balance….

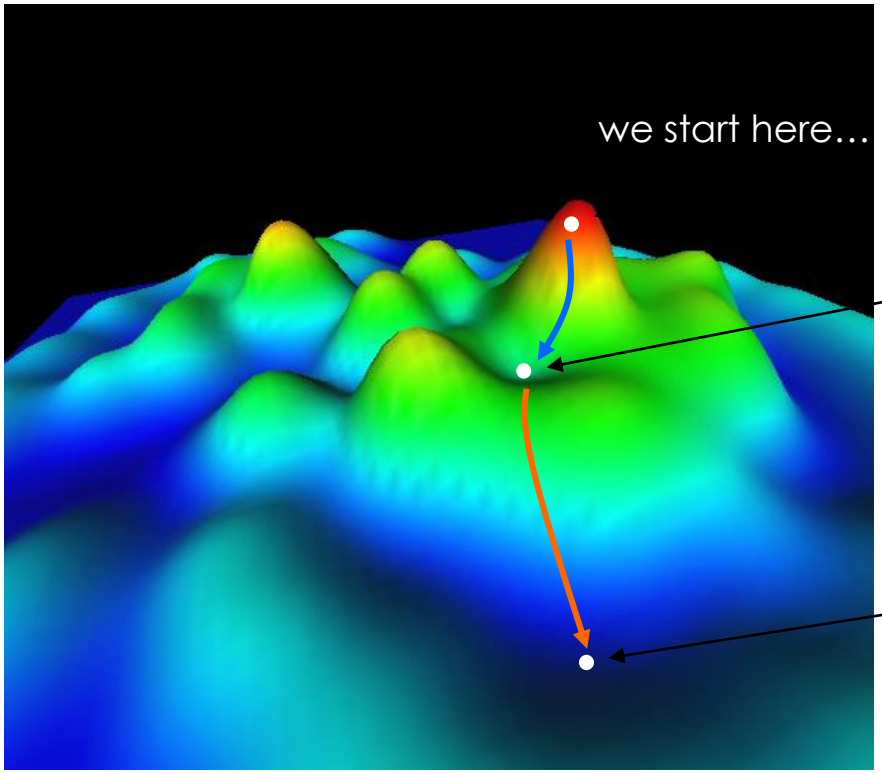


Force-Directed Layouts

Problems:

- the energy function is not monotonic!
- minimizers such as previous gradient descent work locally → can get stuck in local minima
- solve this by more advanced minimizers (see e.g. [Di Battista et al ' 94])
- drawings not intuitive; no clear ordering → where to start reading the drawing??

energy 2D height plot

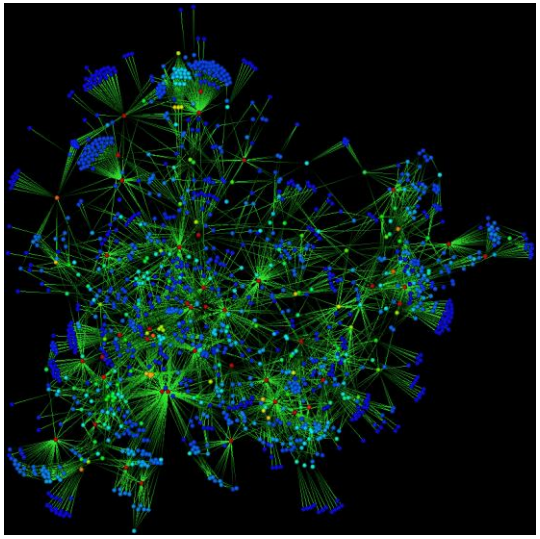


G. Di Battista, P. Eades, J. Tollis, Graph Drawing, Addison-Wesley, 1994

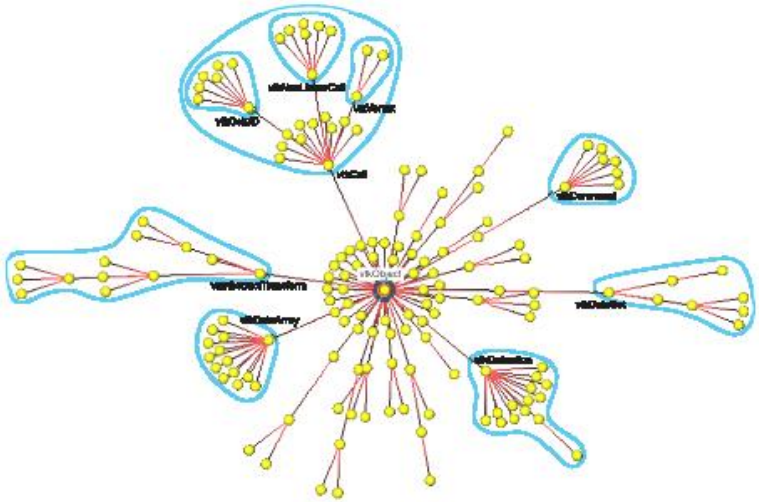
Force-Directed Layouts

Examples

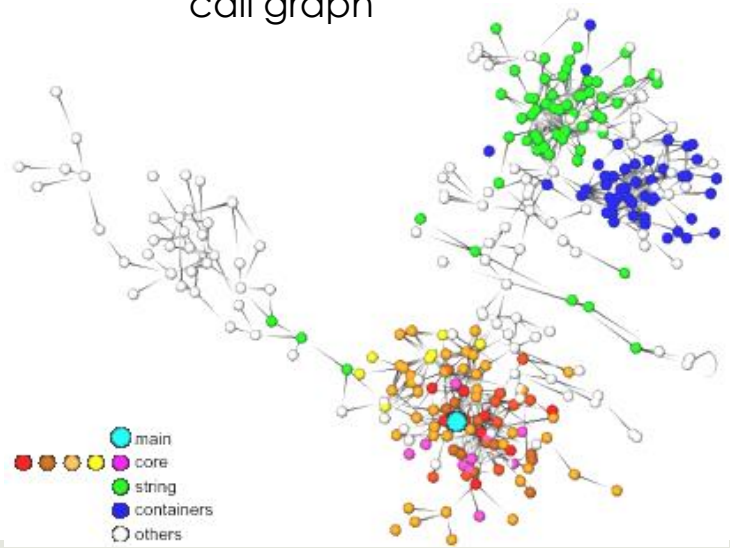
call graph



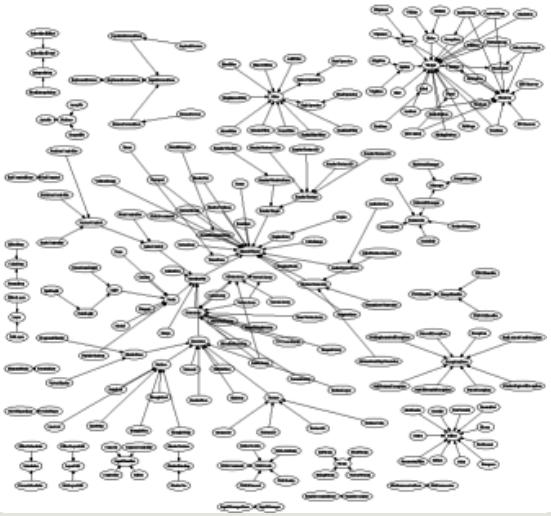
inheritance tree



call graph



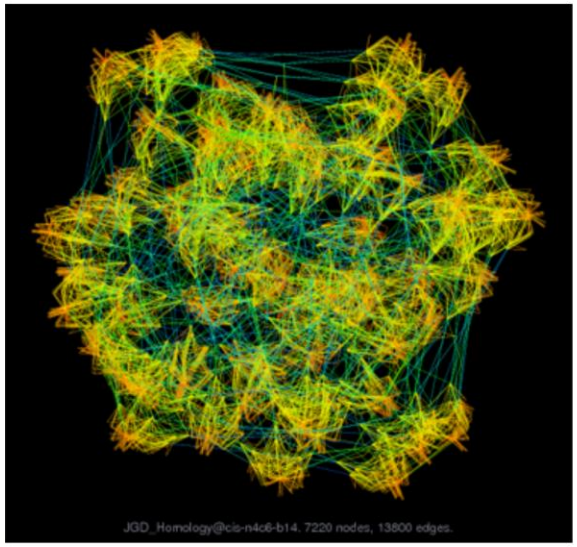
inheritance forest



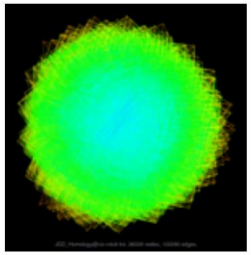
Multi-Level Force-Directed Graph Layout

Idiom: **sfdp** (multi-level force-directed placement)

- data
  - original: network
  - derived: cluster hierarchy atop it
- considerations
  - better algorithm for same encoding technique
    - same: fundamental use of space
    - hierarchy used for algorithm speed/quality but not shown explicitly
- scalability
  - nodes, edges: 1K-10K
  - hairball problem eventually hits



[Efficient and high quality force-directed graph drawing. Hu. The Mathematica Journal 10:37–71, 2005.]

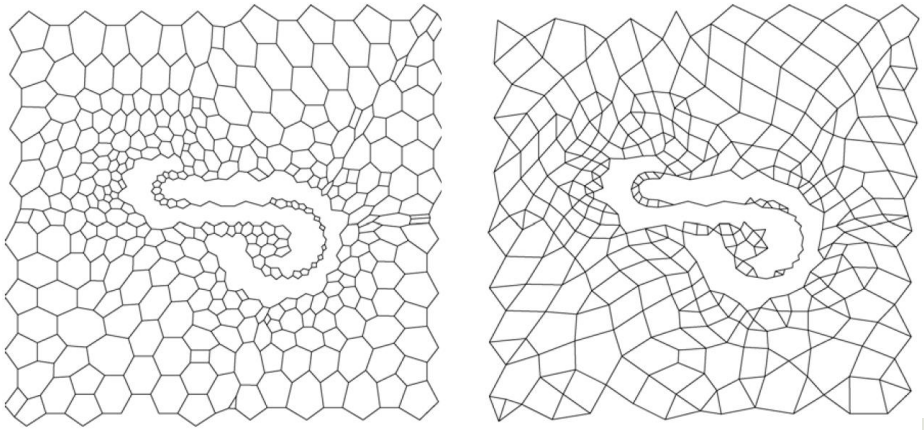


http://www.research.att.com/yifanhu/GALLERY/GRAPHS/index1.html

Multi-Level Force-Directed Graph Layout

The multilevel approach has three distinctive phases: coarsening, coarsest graph layout, and prolongation and refinement. In the coarsening phase, a series of coarser and coarser graphs, G^0 , G^1 , … , G^l , are generated. The aim is for each coarser graph G^{k+1} to encapsulate the information needed to lay out its "parent" G^k while containing fewer vertices and edges. The coarsening continues until a graph with only a small number of vertices is reached. The optimal layout for the coarsest graph can be found cheaply. The layouts on the coarser graphs are recursively prolonged to the finer graphs, with further refinement at each level.

- *Efficient, high-quality force-directed graph drawing* by Hu

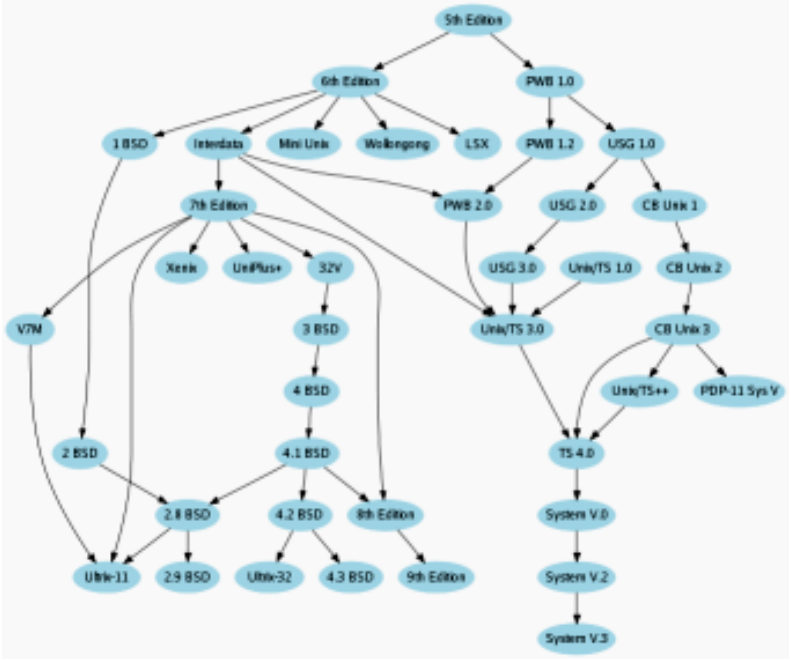


Graph coarsening example

Directed Acyclic Graphs

- class hierarchies (multiple inheritance)
- organization structure (multiple bosses)
- also created from general graphs by removing cycles

Hierarchical layout [Sugiyama et al '81]



UNIX system history drawn with the GraphViz package (www.graphviz.org)

Algorithm:

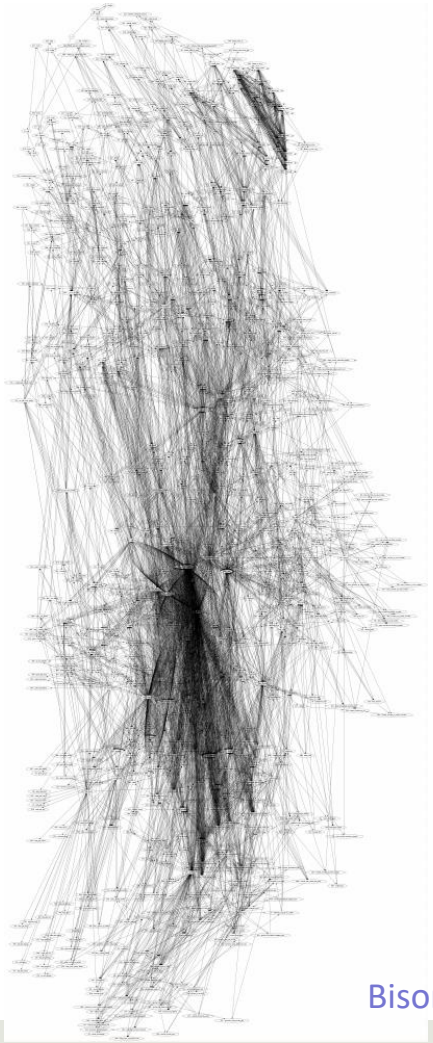
- swap edges to eliminate cycles and get a directed (rooted) graph
- for every level, starting from root:
  - assign y coordinate as function of level
  - permute nodes on level to minimize edge lengths/crossings
- edges drawn as curves (splines) to minimize crossings

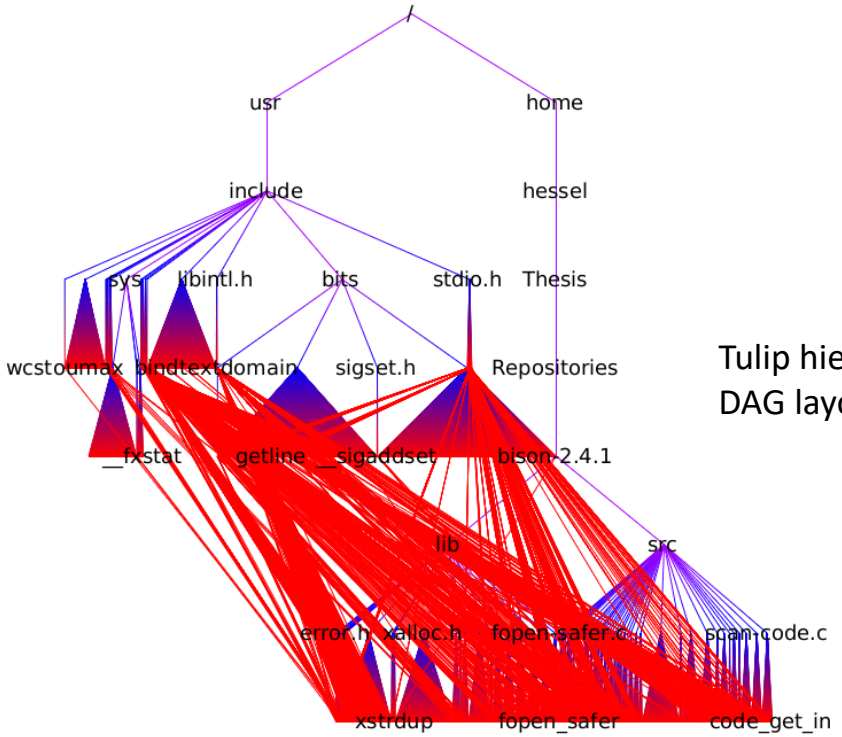
Directed Acyclic Graphs

Hierarchical layout scalability

- OK for < 1000 nodes or edges
- too many crossings and/or bad aspect ratios for large graphs
- we shall see later how to handle large graphs

GraphViz hierarchical DAG layout





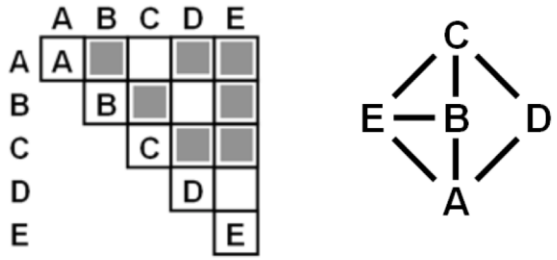
Tulip hierarchical DAG layout

Bison library call graph (3.214 nodes, 14.382 edges)

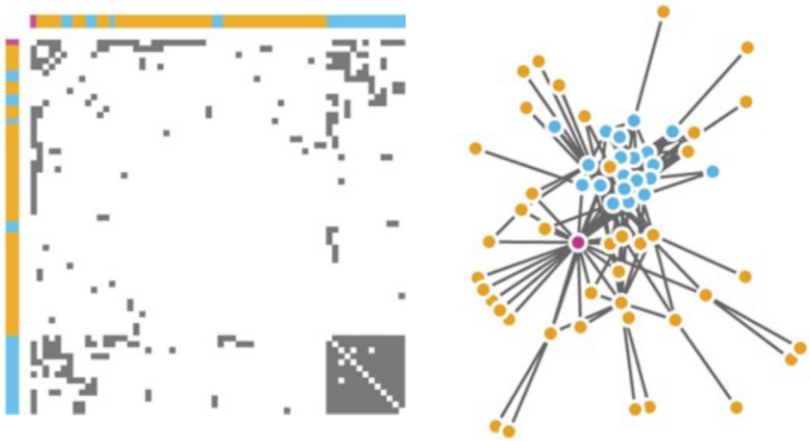
Adjacency Matrix Representation

Idiom: **adjacency matrix view**

- data: network
  - transform into same data/encoding as heatmap
- derived data: table from network
  - 1 quant attrib
    - weighted edge between nodes
  - 2 categ attribs: node list x 2
- visual encoding
  - cell shows presence/absence of edge
- scalability
  - 1K nodes, 1M edges

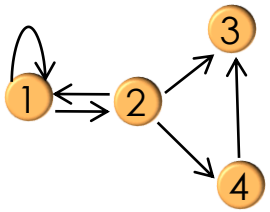


[NodeTrix: a Hybrid Visualization of Social Networks. Henry, Fekete, and McGuffin. IEEE TVCG (Proc. InfoVis) 13(6):1302-1309, 2007.]



[Points of view: Networks. Gehlenborg and Wong. Nature Methods 9:115.]

Matrix Representation



graph

ingoing edges

	1	2	3	4
1	X	X		
2	X			
3		X		X
4		X		

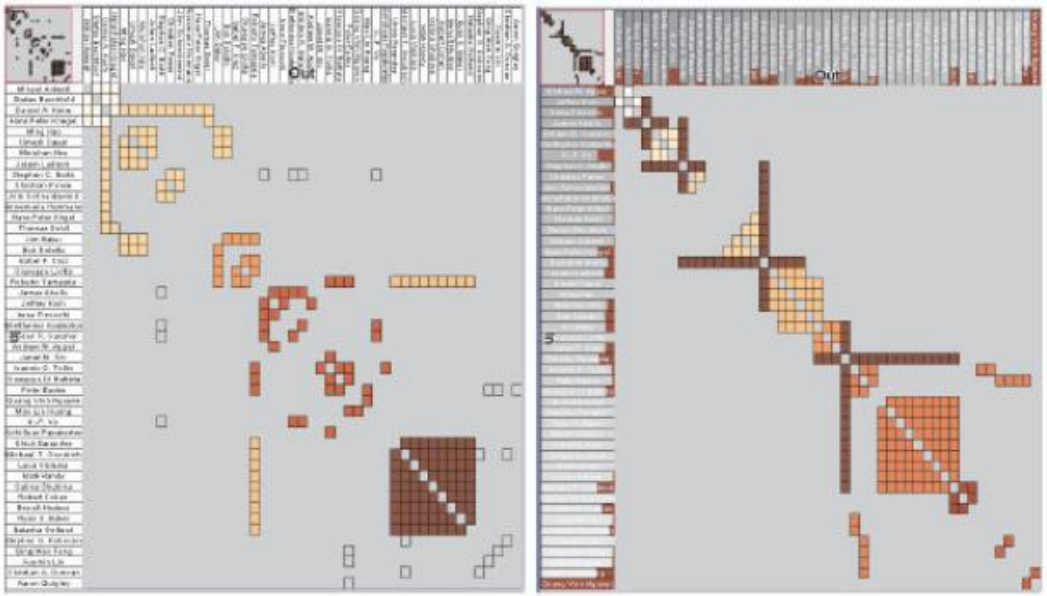
directed adjacency matrix

	1	2	3	4
1	X	X		
2	X		X	X
3		X		X
4		X	X	

undirected adjacency matrix

social network graph

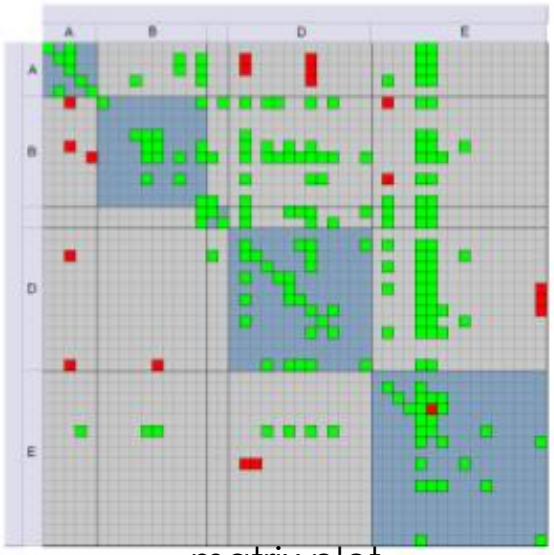
- order of rows and columns highly impacts the visualization, e.g. spotting clusters



N. Henry, J. D Fekete: MatrixExplorer: A Dual-representation System to Explore Social Networks, IEEE TVCG, 2007

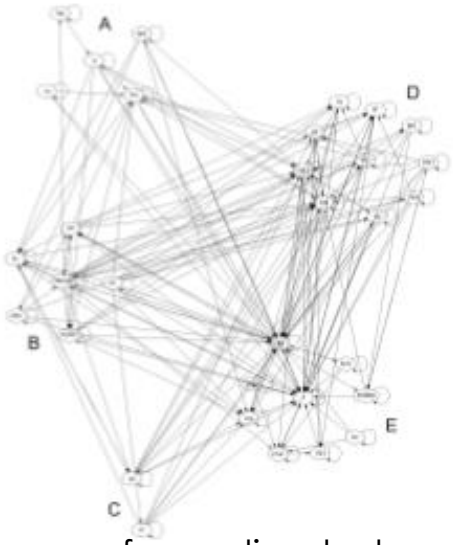
Application: Visualizing Software Structure

- consider a software call graph
  - nodes: functions; edges: call relations (f calls g – directed graph)
  - functions are contained in modules ($f \in A, g \in B, \ldots$)
  - some calls are allowed, some now (depending on system architecture)
- which layout is best for spotting call structure problems?



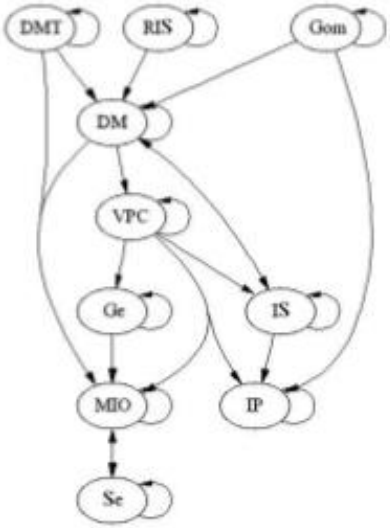
matrix plot

- no clutter, clean image
- show violations by colors (red=not allowed)
- tracing call paths is hard
- detecting 'connected components' is hard



force directed

- some connected components visible
- image is cluttered…
  …even if this graph is really small
- least preferred option



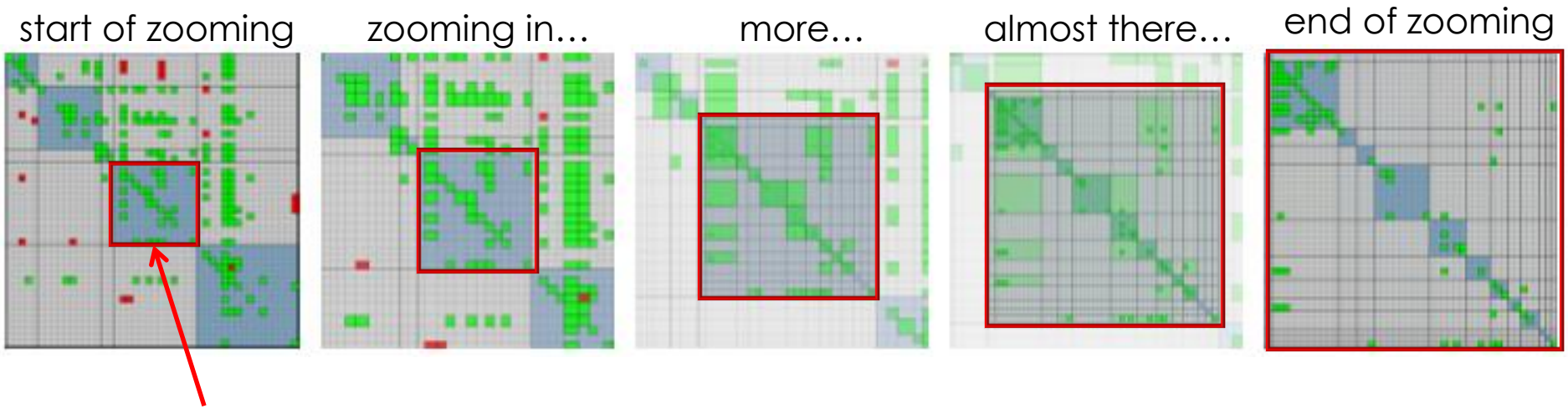
hierarchical DAG
(for a subgraph only)

- good for path tracing
- some structure is apparent…
  …but is it a layout artefact?
- top-down order meaningful

Matrix Plots

Interaction

- zoom in and out – display the matrix at different levels of detail
- smoothly interpolate the cell sizes (using alpha blending) during zooming



F. van Ham, Using Multilevel Call Matrices in Large Software Projects, IEEE InfoVis 2003, www.win.tue.nl/~fvham/matrix

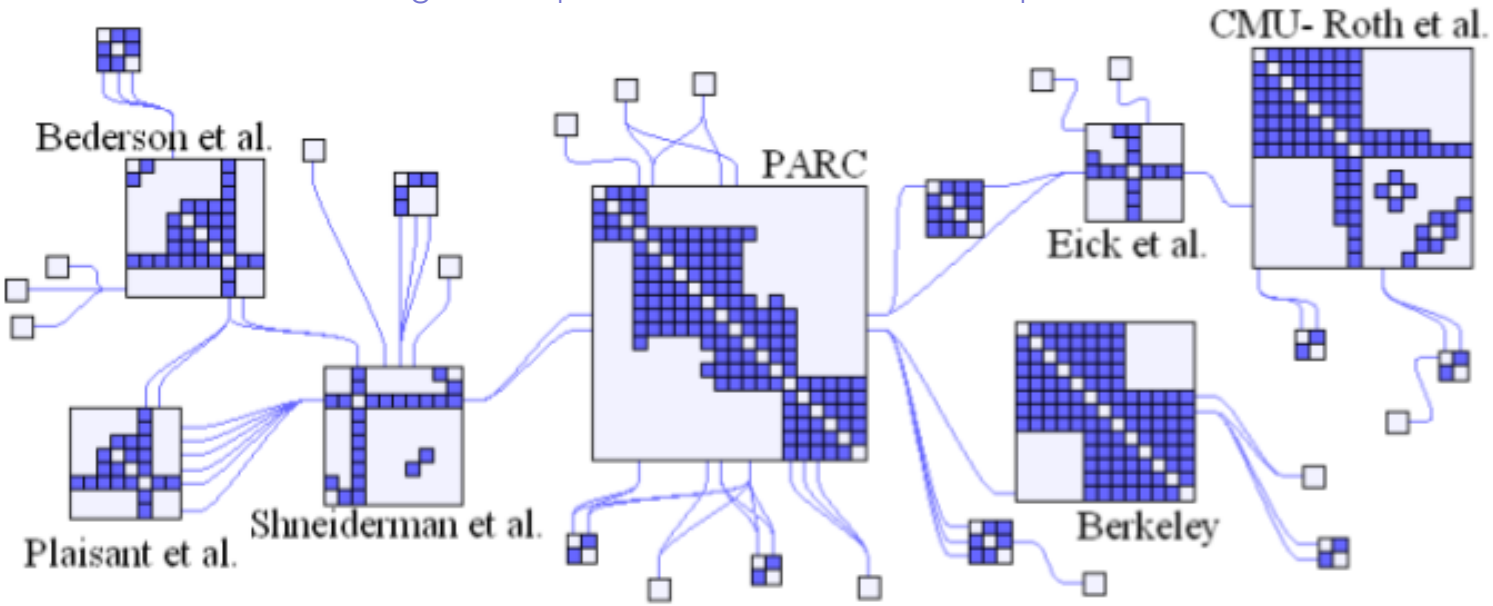
Matrix Plots and Node-Link Layouts

- matrix plots are compact and clean, but don't clearly show structure
- node-link layouts show structure, but get cluttered

Idea

- aggregate graph (find clusters of strongly-connected components)
- show each cluster with a matrix plot
- show cluster-graph with a node-link layout

Largest component of InfoVis co-authorship network

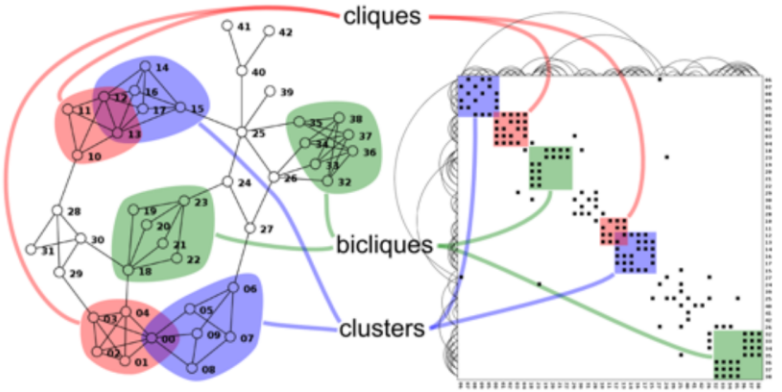


N. Henry, J. D. Fekete, M. McGuffin, NodeTrix: A Hybrid Visualization of Social Networks, IEEE TVCG (InfoVis) 2007

Connection Versus Adjacency Comparison

- adjacency matrix strengths
  - predictability, scalability, supports reordering
  - some topology tasks trainable
- node-link diagram strengths
  - topology understanding, path tracing
  - intuitive, no training needed
- empirical study
  - node-link best for small networks
  - matrix best for large networks
    - if tasks don't involve topological structure!

[On the readability of graphs using node-link and matrix-based representations: a controlled experiment and statistical analysis. Ghoniem, Fekete, and Castagliola. Information Visualization 4:2 (2005), 114–135.]



http://www.michaelmcguffin.com/courses/vis/patternsInAdjacencyMatrix.png

Radial Tree Layout

Idiom: **radial node-link tree**

- data
  - tree
- encoding
  - link connection marks
  - point node marks
  - radial axis orientation
    - angular proximity: siblings
    - distance from center: depth in tree
- tasks
  - understanding topology, following paths
- scalability
  - 1K - 10K nodes

http://mbostock.github.com/d3/ex/tree.html

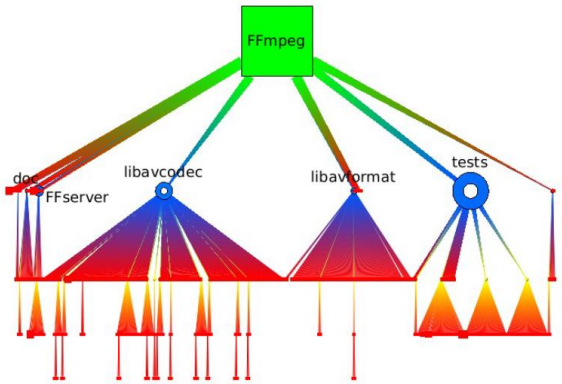
Tree Layout Idioms

Trees are acyclic graphs

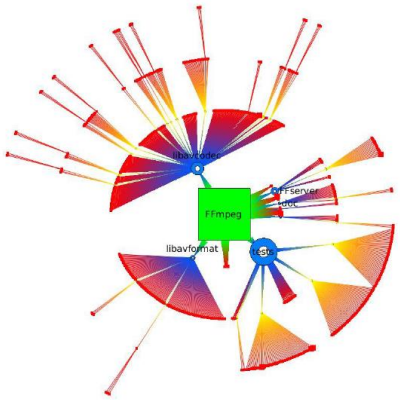
Examples

- icon size: folder size
- icon shape+color: level in tree

rooted tree

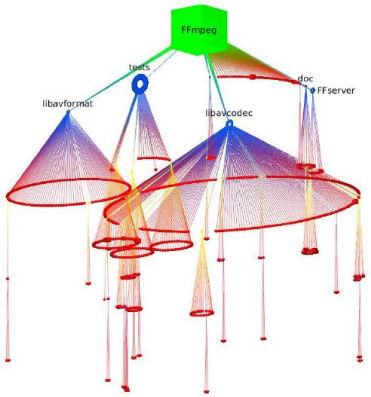


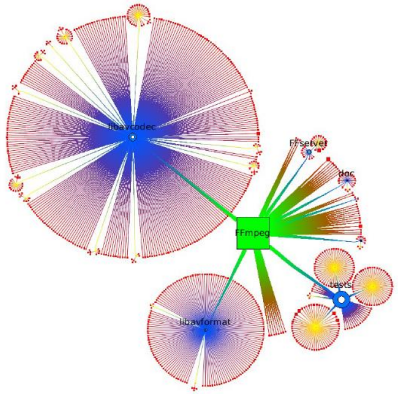
ffmpeg video code C library: 785 files, 42 folders



radial tree

cone tree

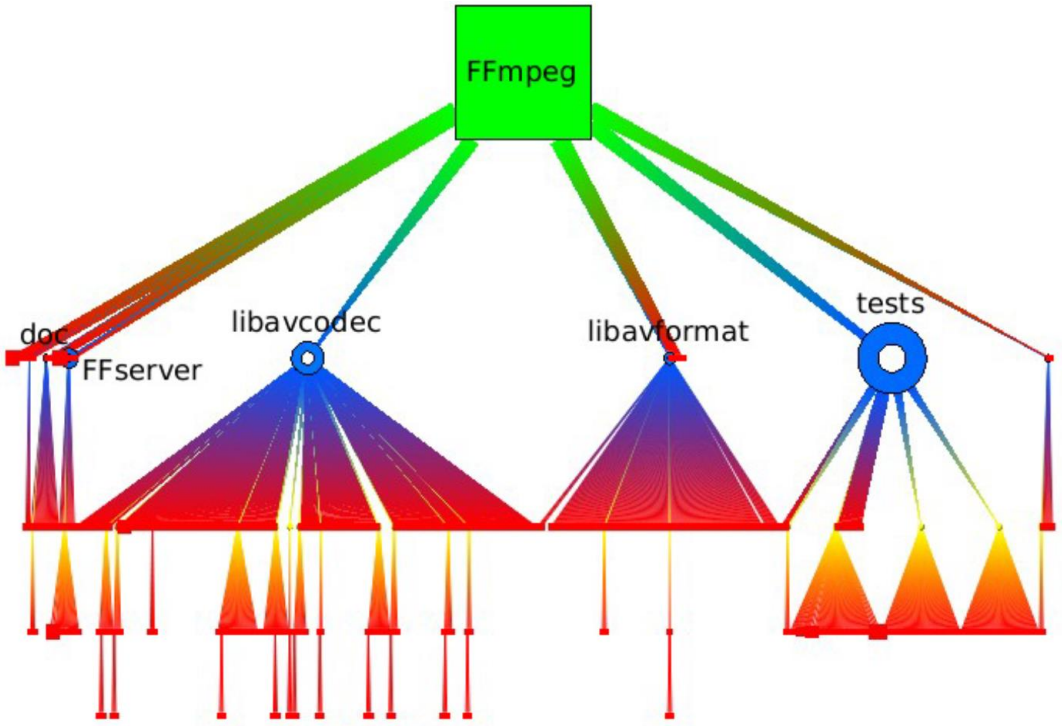




bubble tree

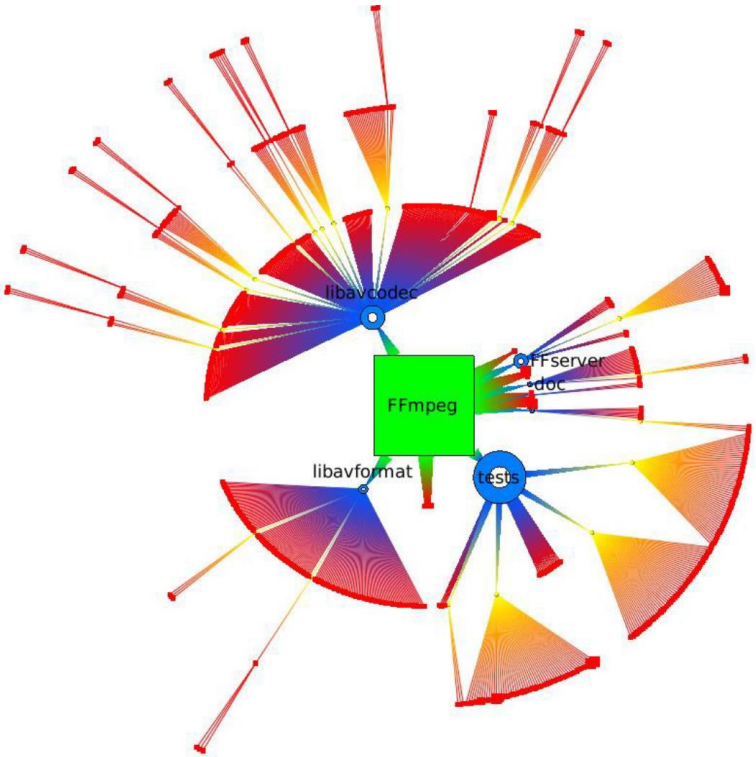
root
level 2
level 3
others

Tree Layout: Rooted Layout



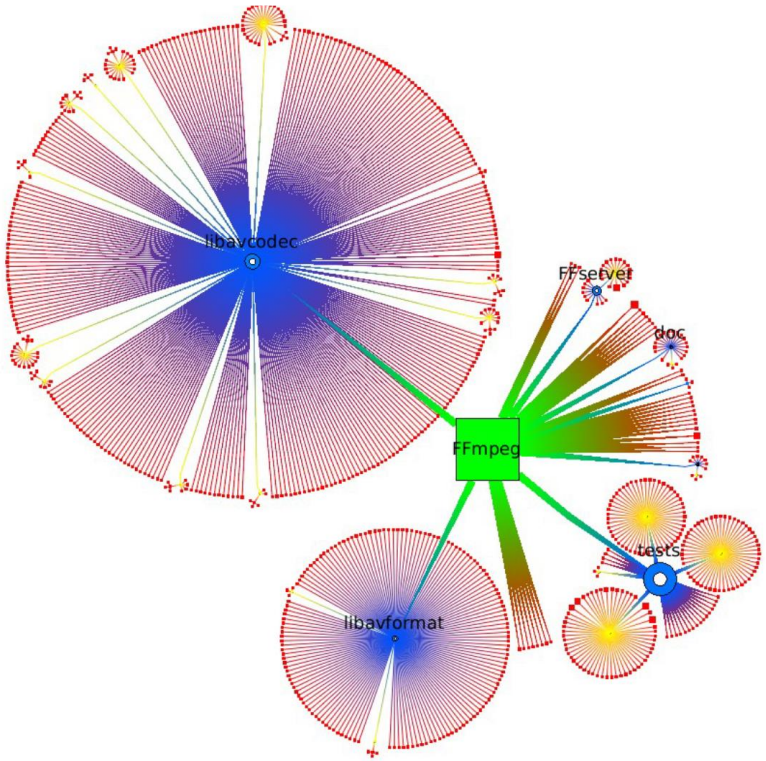
- very familiar to virtually everybody
- size (# children) and depth of sub-trees easy to perceive
- smooth edge shading → emphasize colors of small nodes
- unbalanced aspect ratios can occur → limited scalability

Tree Layout: Radial Layout



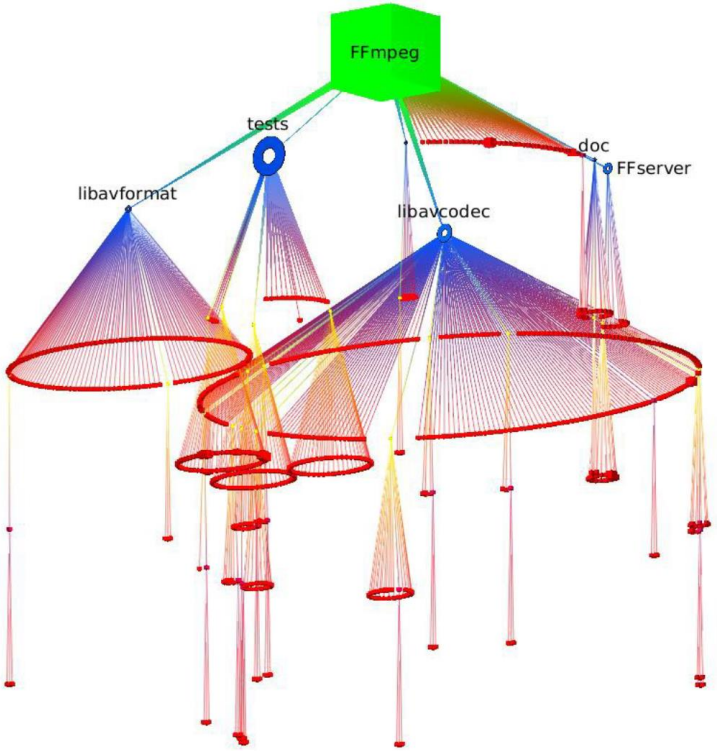
- like rooted tree, but arc-length used instead of x axis
- size (# children) and depth of sub-trees easy to perceive
- good aspect ratio guaranteed
- nodes close to root get less space

Tree Layout: Bubble Layout



- a subtree gets a full circle instead of a circle sector
- better spreading of the nodes for large trees
- variable edge lengths
- hard to distinguish node depth in the tree

Cone Tree

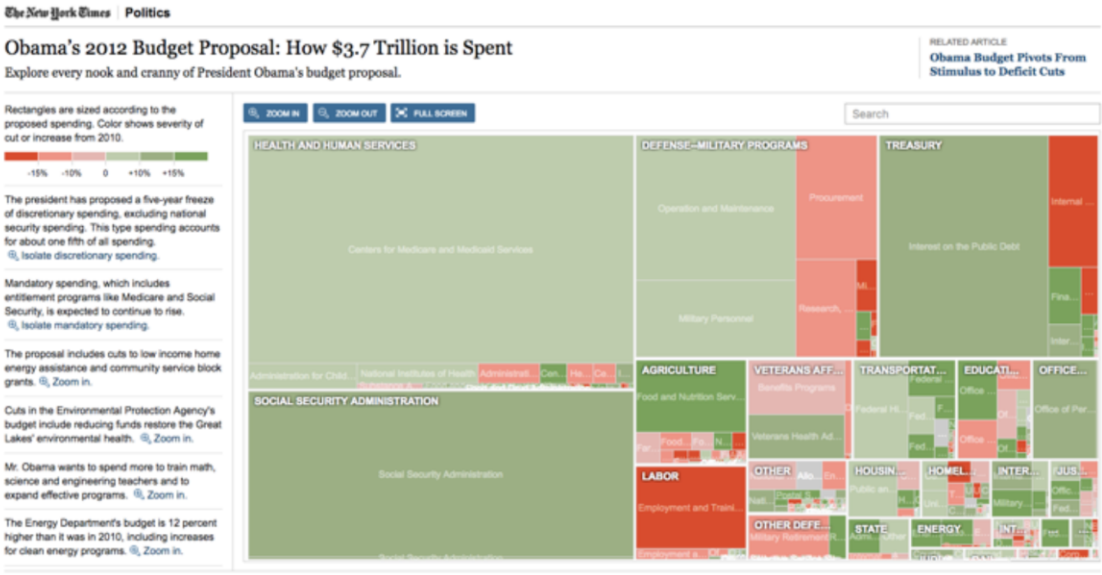


- a subtree gets a full cone instead of a sector / circle / line
- 3D effectively shows the tree depth
- combines bubble tree (seen from above) with rooted tree (seen from profile)
- 3D is tricky: occlusions, perspective shortening, navigation

TreeMaps

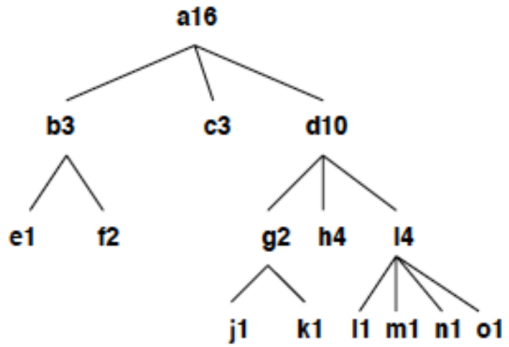
Idiom: **treemap**

- data
  - tree
  - 1 quant attrib at leaf nodes
- encoding
  - area containment marks for hierarchical structure
  - rectilinear orientation
  - size encodes quant attrib
- tasks
  - query attribute at leaf nodes
- scalability
  - 1M leaf nodes

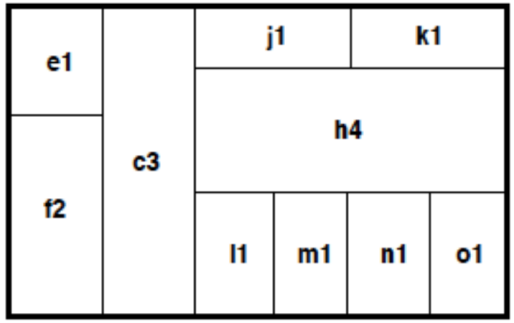


http://www.nytimes.com/packages/html/newsgraphics/2011/0119-budget/index.html

TreeMap Layout Algorithm



(a) Tree diagram

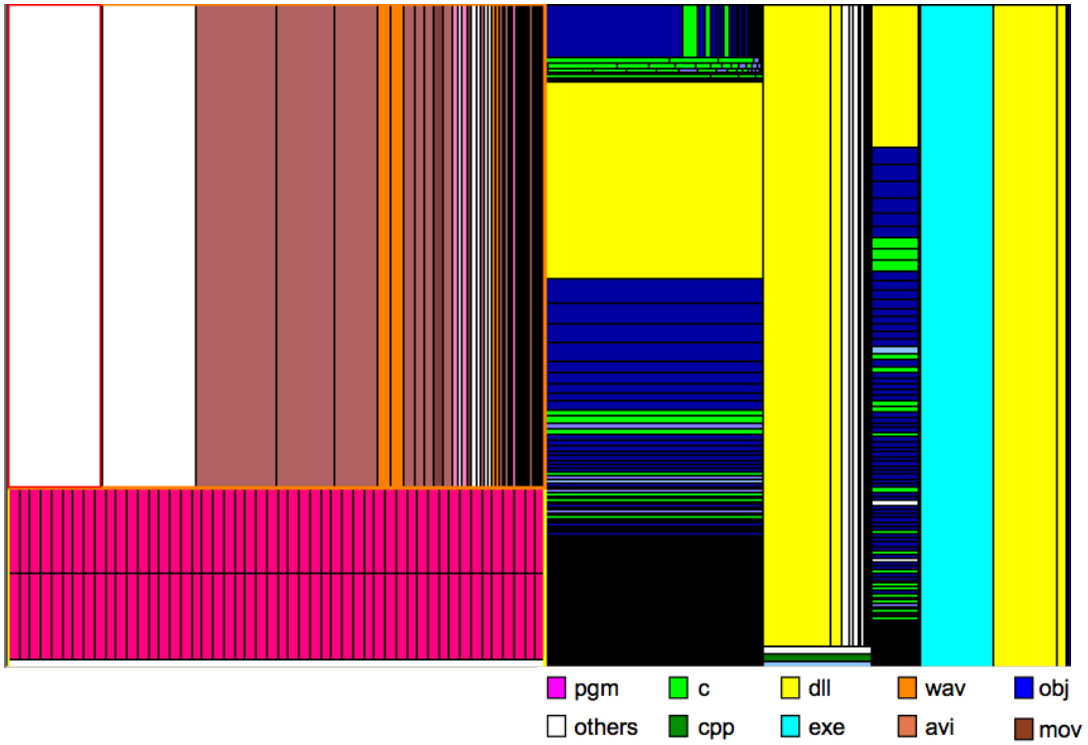


(b) Treemap

Treemap "Slice and Dice" Layout Algorithm

- Alternate the axis you divide each level
- Divide the axis into lengths based on a metric..
  - Examples: uniform, number of nodes in subtree, application specific importance
- Recurse until all the nodes have been processed
- Only leaves explicitly visisble

Treemaps

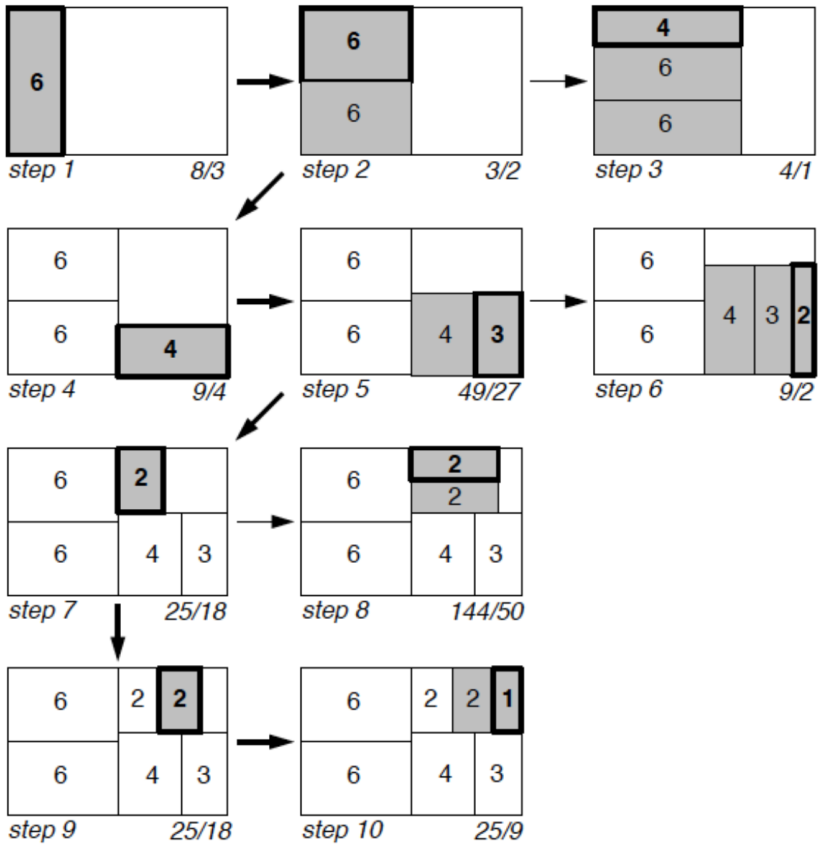


ffmpeg C library

Basic idea: 'slice and dice' layout

- 1 node = 1 rectangle
- child node rectangles: *nested* in the parent node rectangle (recursive subdivisio
- leaf rectangle *size* and *color* show data attributes
- edges: not drawn explicitly!
- very compact: tens of thousands of nodes on one screen
- hierarchy depth unclear

Squarified Treemap Layout



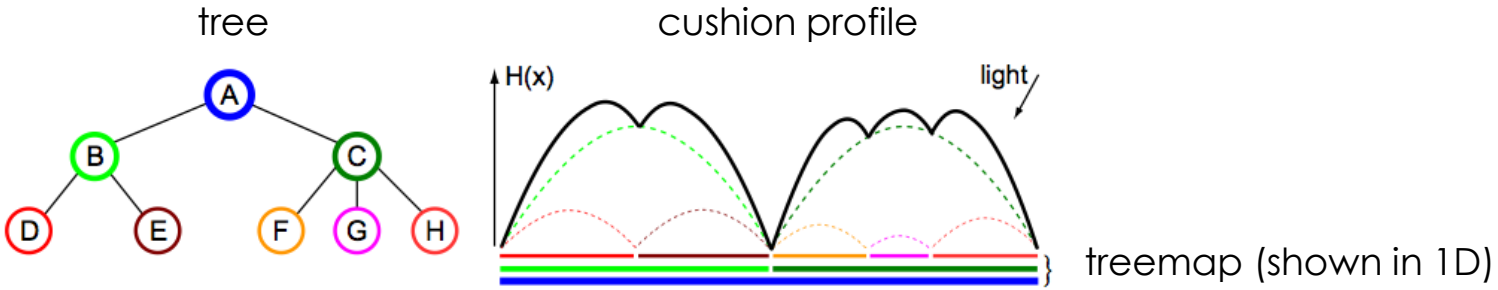
Treemap "Squarified" Layout Algorithm

- Start with list of child nodes
- Sort by area, largest first
- Split the longest axis
- Start filling one "half" in alternate direction of the split (step 2)
  - Keep filling until aspect ratios get worse (step 3)
- Repeat process on the unfilled portion of the original

Squarified TreeMaps

Two extensions:

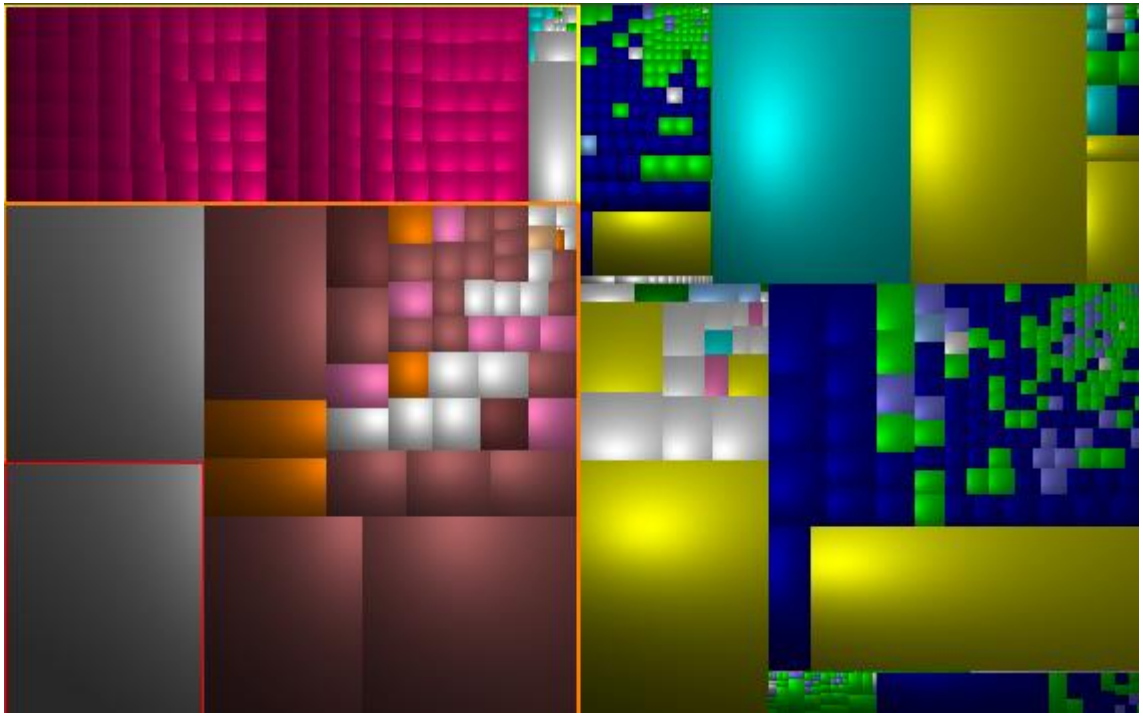
- enforce near-square aspect ratios during rectangle subdivision [Bruls et al '00]
- use shading: cushion profiles to convey hierarchical structure [Van Wijk et al '99]



treemap (shown in 1D)

- for each rectangle r_i (except root)
  - define 2D parabola $h_i(x,y)$ with height $H_i = f^d$ (d = depth of r_i in tree)
- compute global height $h = \Sigma_i h_i$
- for each image pixel $I(x,y)$
  - $I(x,y) = shading\ (h(x,y))$
  - use an oblique light source for better results

Squarified Cushion Treemaps



ffmpeg C library

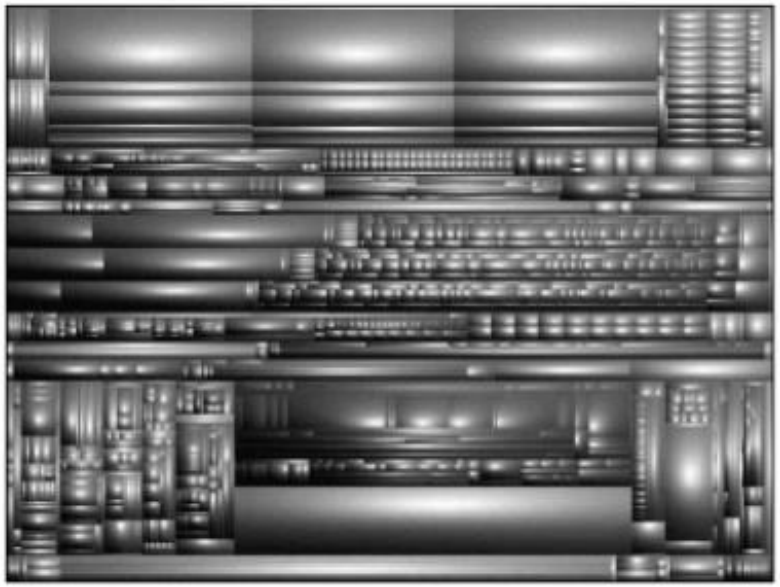
- rectangle borders are not *explicitly* drawn → gain space
- borders are *implicit* in the shading discontinuities
- discontinuity strength conveys tree depth levels

Image generated with the SequoiaView tool (www.win.tue.nl/sequoiaview)

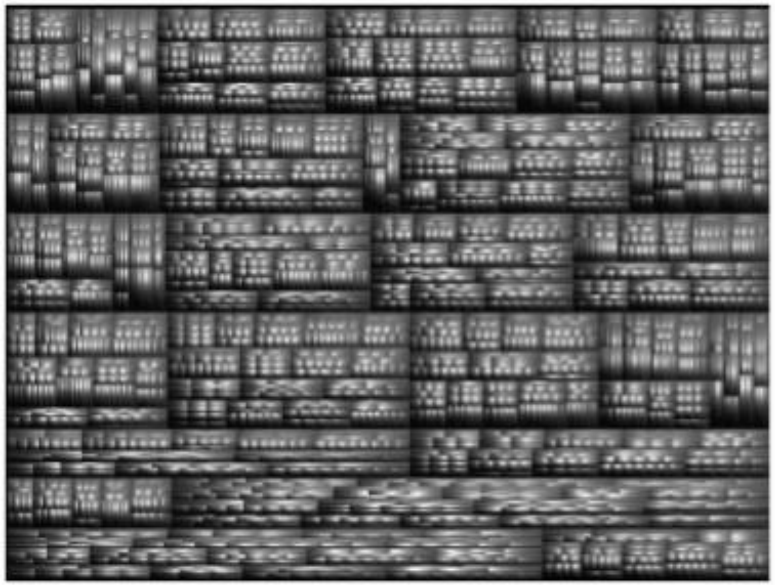
Squarified Cushion Treemaps

Comparison of methods

hard disk, ~30K files



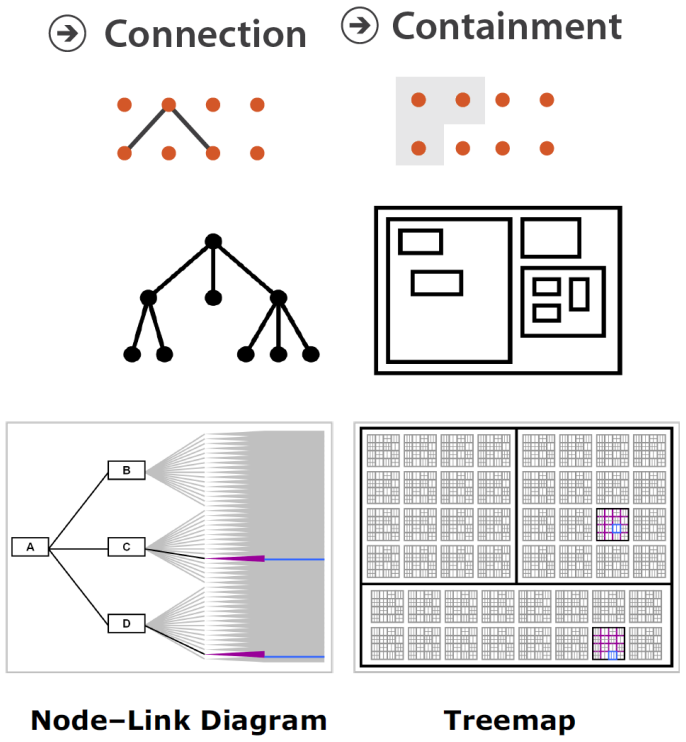
- slice and dice layout
- unbalanced sizes
- structure is not very clear



- squarified layout
- balanced sizes
- structure is extremely salient

Link Marks: Connection and Containment

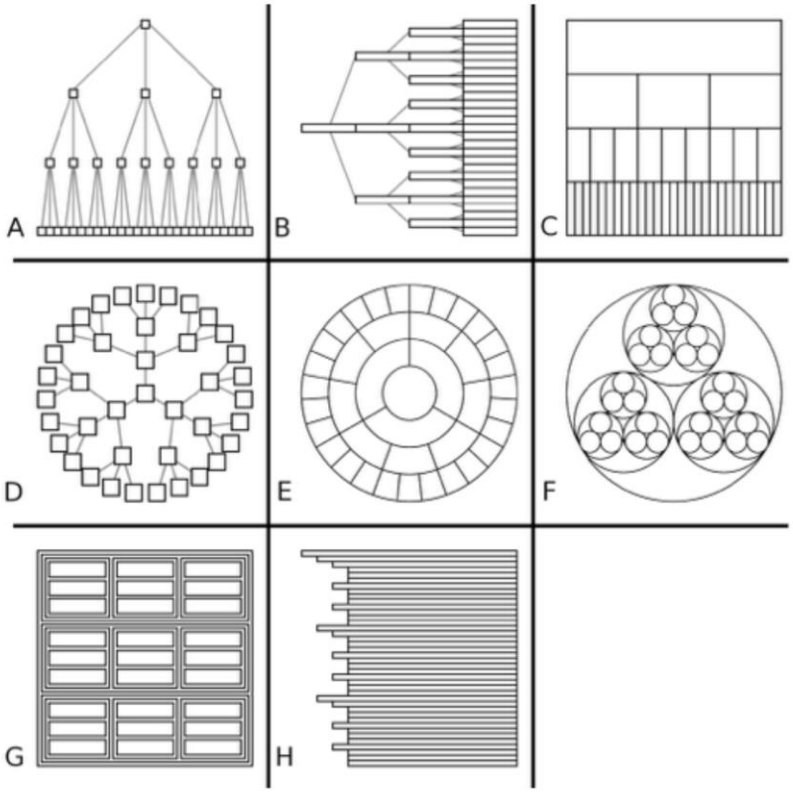
- marks as links (vs. nodes)
  - common case in network drawing
  - 1D case: connection
    - ex: all node-link diagrams
    - emphasizes topology, path tracing
    - networks and trees
  - 2D case: containment
    - ex: all treemap variants
    - emphasizes attribute values at leaves (size coding)
    - only trees



[Elastic Hierarchies: Combining Treemaps and Node-Link Diagrams. Dong, McGuffin, and Chignell. Proc. InfoVis 2005, p. 57-64.]

Tree Drawing Comparisons

- data shown
  - link relationships
  - tree depth
  - sibling order
- design choices
  - connection vs containment link marks
  - rectilinear vs radial layout
  - spatial position channels
- considerations
  - redundant? arbitrary?
  - information density?
    - avoid wasting space



[Quantifying the Space-Efficiency of 2D Graphical Representations of Trees. McGuffin and Robert. Information Visualization 9:2 (2010), 115–140.]