TREES & GRAPHS

Alex Bigelow
University of Utah
TREES & GRAPHS

Alex Bigelow
University of Utah

slide acknowledgements:
Miriah Meyer, University of Utah
Hanspeter Pfister, Harvard University
Jeff Heer, Stanford University
administrivia . . .
exam 1
EXAM REVIEW
last time . . .
**Arrange Tables**

- Express Values

**Separate, Order, Align Regions**

- Separate
- Order
- Align

- 1 Key List
- 2 Keys Matrix
- 3 Keys Volume
- Many Keys Recursive Subdivision

**Axis Orientation**

- Rectilinear
- Parallel
- Radial

**Layout Density**

- Dense
- Space-Filling
today . . .
dataset types

Tables
- Items
- Attributes

Networks & Trees
- Items (nodes)
- Links
- Attributes

Fields
- Grids
- Positions
- Attributes

Geometry
- Items
- Positions

Clusters, Sets, Lists
- Items

Multidimensional Table

Trees

Grid of positions
- Attributes (columns)
- Value in cell
www.cs.utah.edu

- **blue**: for links (the A tag)
- **red**: for tables (TABLE, TR and TD tags)
- **green**: for the DIV tag
- **violet**: for images (the IMG tag)
- **yellow**: for forms (FORM, INPUT, TEXTAREA, SELECT and OPTION tags)
- **orange**: for linebreaks and blockquotes (BR, P, and BLOCKQUOTE tags)
- **black**: the HTML tag, the root node
- **gray**: all other tags
Webpages as Graphs
DEFINITIONS
- A graph $G$ consists of a collection of vertices (or nodes) $V$ and a set of edges $E$, consisting of vertex pairs.

- An edge $e_{xy} = (x,y)$ connects two vertices $x$ and $y$.
  - For example: $V = \{1,2,3,4\}$, $E = \{(1,2),(1,3),(2,3),(3,4),(4,1)\}$
### Nodes

<table>
<thead>
<tr>
<th>ID</th>
<th>Attribute 1</th>
<th>Attribute 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3.4</td>
<td>Good</td>
</tr>
<tr>
<td>2</td>
<td>5.8</td>
<td>Bad</td>
</tr>
<tr>
<td>3</td>
<td>1.1</td>
<td>Ugly</td>
</tr>
<tr>
<td>4</td>
<td>-3.5</td>
<td>Really Ugly</td>
</tr>
</tbody>
</table>

### Edges

<table>
<thead>
<tr>
<th>Source</th>
<th>Target</th>
<th>Attribute 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>100</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>200</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>50</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>150</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>250</td>
</tr>
</tbody>
</table>
a bunch of definitions

- A directed graph
- An undirected graph
- Weighted
- Unconnected
- Node degrees
- A cycle
- An acyclic graph
- A connected acyclic graph, a.k.a. a tree
- A rooted tree or hierarchy
- Node depths
GRAPHS & TREES

- graphs
  - model relations amount data
  - nodes and edges

- trees
  - graphs with hierarchical structure
  - nodes as parents and children
VISUALIZING TREES
ROOTED TREES

- recursion makes it elegant and fast to draw trees

- approaches:
  - node link
  - layered
  - indentation
  - enclosure
NODE-LINK DIAGRAMS

- nodes are distributed in space, connected by straight or curved lines
- typical approach is to use 2D space to break apart breadth and depth
- often space is used to communicate hierarchical orientation
LAYERED DIAGRAMS

- recursive subdivision of space
- structure encoded using:
  - layering
  - adjacency
  - alignment
SCALE PROBLEM

- tree breadth often grows exponentially
- quickly run out of space!

- solutions
  - scrolling or panning
  - filtering or zooming
  - hyperbolic layout
INDENTATION

- place all items along vertically spaced rows
- indentation used to show parent/child relationships
- commonly used as a component in an interface
- breadth and depth contend for space
- often requires a great deal of scrolling
ENCLOSURE DIAGRAMS

- **encode structure using spatial enclosure**
  - often referred to as *treemaps*

- **benefits**
  - provides single view of entire tree
  - easier to spot small / large nodes

- **problems**
  - difficult to accurately read depth
TREEMAPS

- recursively fill space based on a size metric for nodes
- enclosure indicates hierarchy
- additional measures can control aspect ratio of cells
- most often use rectangles, but other shapes possible
  - square, circle, voronoi tessellation
VISUALIZING GRAPHS
VISUALIZING GRAPHS

- node link layouts
  - Reingold-Tilford (trees only)
  - Sugiyama (directed acyclic graphs)
  - Force directed
  - Attribute-based

- adjacency matrices

- aggregate views
  - Motif Glyphs
  - PivotGraph
SPATIAL LAYOUT

- primary concern of graph drawing is the spatial layout of nodes and edges
- often (but not always) the goal is to effectively depict the graph structure
  - connectivity, path-following
  - network distance
  - clustering
  - ordering (e.g., hierarchy level)
REINGOLD-TILFORD

- repeatedly divide space for subtrees by leaf count
  - breadth of tree along one dimension
  - depth along the other dimension
REINGOLD-TILFORD

- **goal**
  - make smarter use of space
  - maximize density and symmetry

- **design concerns**
  - clearly encode depth level
  - no edge crossings
  - isomorphic subtrees drawn identically
  - compact

- **approach**
  - bottom up recursive approach
  - for each parent make sure every subtree is drawn
  - pack subtrees as closely as possible
  - center parent over subtrees
SUGIYAMA

- great for graphs that have an intrinsic ordering
- depth not strictly encoded
- What is the depth of V7M?

UNIX ancestry
SUGIYAMA

+ nice, readable top down flow

+ relatively fast (depending on heuristic used for crossing minimization)

- not really suitable for graphs that don’t have an intrinsic top down structure

- hard to implement
  - use free graphviz lib instead: http://www.graphviz.org
FORCE-DIRECTED

- no intrinsic layering, now what?

- physics model
  - edges = springs
  - nodes = repulsive particles
FORCE MODEL

- many variations, but usually physical analogy:
  - repulsion: \( f_R(d) = C_R \times m_1 \times m_2 / d^2 \)
    - \( m_1, m_2 \) are node masses
    - \( d \) is distance between nodes
  - attraction: \( f_A(d) = C_A \times (d - L) \)
    - \( L \) is the rest length of the spring
    - i.e. Hooke’s Law

- total force on a node \( x \) with position \( x' \)
  \( \sum_{\text{neighbors}(x)} f_A(||x'-y'||) \times (x'-y') + -f_R(||x'-y'||) \times (x'-y') \)
ALGORITHM

- start from random layout

- (global) loop:
  - for every node pair compute repulsive force
  - for every edge compute attractive force
  - accumulate forces per node
  - update each node position in direction of accumulated force

- stop when layout is ‘good enough’
FORCE DIRECTED

+ very flexible, aesthetic layouts on many types of graphs
+ can add custom forces
+ relatively easy to implement

- repulsion loop is $O(n^2)$ per iteration
  - can speed up to $O(N \log N)$ using quadtree or k-d tree

- prone to local minima
  - can use simulated annealing
Les Misérables
color co-occurrence

http://bl.ocks.org/mbostock/4062045
OTHER LAYOUTS

- **orthogonal**
  - great for UML diagrams
  - algorithmically complex

- **circular layouts**
  - emphasizes ring topologies
  - used in social network diagrams

- **nested layouts**
  - recursively apply layout algorithms
  - great for graphs with hierarchical structure
The Open Graph Viz Platform

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

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

Gephi 0.8 beta has been released! Discover a new Preview and dynamic features, start building commercial applications with the new open source license.

APPLICATIONS

- Exploratory Data Analysis: intuition-oriented analysis by networks manipulations in real time.
- Link Analysis: revealing the underlying structures of associations between objects, in particular in scale-free networks.
- Social Network Analysis: easy creation of social data connectors to map community organizations and small-world networks.
- Biological Network analysis: representing patterns of biological data.
- Poster creation: scientific work promotion with hi-quality printable maps.

“Like Photoshop™ for graphs.” — the Community

PAPERS

LATEST NEWS
- Weekly news February 27, 2012
- Annual report 2011 February 25, 2012
- Gephi-Neo4j presentation at FOSEM February 20, 2012
- Gephi meet-up #4 in Berlin February 2, 2012
- Introducing the Gephi Plugins Bootcamp January 12, 2012
ATTRIBUTE-DRIVEN LAYOUT

- large node-link diagrams get messy!
- are there additional structures we can exploit?

- idea: use data attributes to perform layout
  - e.g., scatterplot based on node values
- dynamic queries and/or brushing can be used to enhance perception of connectivity
cerebral

Barsky 2008
NODE LINK

+ understandable visual mapping

+ can show overall structure, clusters, paths

+ flexible, many variations

- all but the most trivial algorithms are $O(N^2)$

- not good for dense graphs
  - hairball problem!
ALTERNATIVE: ADJACENCY MATRIX

- instead of node link diagram, use adjacency matrix representation
SPOTTING PATTERNS IN MATRICES

Henry 2006
Les Misérables
classification
co-occurrence

http://bost.ocks.org/mike/miserables/
+ great for dense graphs

+ visually scalable

+ can spot clusters

- row order affects what you can see

- abstract visualization

- hard to follow paths
MOTIF GLYPHS

Connector

Fan

Clique

Dunne 2013
MOTIF GLYPHS

Dunne 2013
MOTIF GLYPHS

Dunne 2013
CRITIQUE

- When *can* you use this technique?
- When *should* you use this technique?
PIVOT GRAPHS

- new graph, derived from categorical node attributes
- 1D or 2D layouts possible
- size of nodes and edges related to number of aggregated original nodes and edges
- scalability through abstraction, not layout algorithm
CRITIQUE

- When *can* you use this technique?
- When *should* you use this technique?
exercise
**GRAPH DRAWING EXERCISE**

adjacency matrix

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

*create an aesthetically pleasing node-link diagram representation*
create an aesthetically pleasing node-link diagram representation
RECAP
- **TREES**
  - **indentation**
    - *simple, effective for small trees*
  - **node link and layered**
    - *looks good but needs exponential space*
  - **enclosure** *(treemaps)*
    - *great for size related tasks but suffer in structure related tasks*

- **GRAPHS**
  - **node link**
    - *familiar, but problematic for dense graphs*
  - **adjacency matrices**
    - *abstract, hard to follow paths*
  - **aggregation can help**
    - *not always possible, not always appropriate*

- **TAKE HOME MESSAGE:** *no best solution*
L14: Text and Sets

REQUIRED READING
CH. 11: INFORMATION VISUALIZATION FOR TEXT ANALYSIS

As discussed in the previous chapter, visualization when applied to text seems to be most effective for specialists doing data analysis. Although this is an exciting field, it is not what most people think of when one talks about search interfaces. Unfortunately, some researchers working on visualization of text conflate search tasks with data analysis tasks. For example, Veerasamy and Heikes, 1997 critique one interface for making it "more difficult than in our tool to gain an overall picture of the query word distribution for a whole set of documents in one glance." It is unclear why a searcher would want to see such a distribution, even though such a view may be of great interest to a computational linguist.
Visualizing Sets and Set-typed Data: State-of-the-Art and Future Challenges

Bilal Alsallakh¹, Luana Micaller²,³, Wolfgang Aigner¹,⁴, Helwig Hauser⁵, Silvia Miksch¹, and Peter Rodgers³

¹Vienna University of Technology, Austria  ²Helsinki Institute for Information Technology HIIT, Finland  ³University of Kent, United Kingdom  ⁴St. Pölten University of Applied Sciences, Austria  ⁵University of Bergen, Norway

Figure 1: Different set visualizations: (a) An Euler diagram [Pod08], (b) Bubble Sets [CPC09], (c) Radial Sets [AAMH13].

Abstract