Graphs and Trees

Graphs
Model relations among data
Nodes and edges

Trees
Graphs with hierarchical structure
Connected graph with N-1 edges
Nodes as parents and children
A primary concern of tree/graph drawing is the spatial arrangement of nodes and edges. Often (but not always) the goal is to effectively depict the graph structure:

- Connectivity, path-following
- Topological distance
- Clustering / grouping
- Ordering (e.g., hierarchy level)
Applications

Tournaments
Organization Charts
Genealogy
Diagramming (e.g., Visio)
Biological Interactions (Genes, Proteins)
Computer Networks
Social Networks
Simulation and Modeling
Integrated Circuit Design
Network Analysis Tasks  [Pretorius '13]

**Structure-based:** relationships and connectivity

**Attribute-based:** specific node/link attributes

**Browsing:** understand paths in the data

**Estimation:** summarization and temporal changes
Network Analysis Tasks [Pretorius '13]

**Structure-based:** relationships and connectivity

Find all of the friends of friends for Taylor.
Find all of the people who are friends with Jordan and Alex.
Six degrees of separation: shortest path between two individuals.

**Attribute-based:** specific node/link attributes

**Browsing:** understand paths in the data

**Estimation:** summarization and temporal changes
Network Analysis Tasks [Pretorius '13]

Structure-based: relationships and connectivity
Find all of the friends of friends for Taylor.
Find all of the people who are friends with Jordan and Alex.
Six degrees of separation: shortest path between two individuals.

Attribute-based: specific node/link attributes
Find all "students" attending CSE442.
Find all the "friends" and "family" of Alex.

Browsing: understand paths in the data

Estimation: summarization and temporal changes
Network Analysis Tasks [Pretorius '13]

**Structure-based:** relationships and connectivity
Find all of the friends of friends for Taylor.
Find all of the people who are friends with Jordan and Alex.
*Six degrees of separation:* shortest path between two individuals.

**Attribute-based:** specific node/link attributes
Find all "students" attending CSE442.
Find all the "friends" and "family" of Alex.

**Browsing:** understand paths in the data
Find Alex's friend Taylor, and then Taylor's friend Jordan.

**Estimation:** summarization and temporal changes
Network Analysis Tasks  [Pretorius '13]

Structure-based: relationships and connectivity
Find all of the friends of friends for Taylor.
Find all of the people who are friends with Jordan and Alex.
Six degrees of separation: shortest path between two individuals.

Attribute-based: specific node/link attributes
Find all "students" attending CSE442.
Find all the "friends" and "family" of Alex.

Browsing: understand paths in the data
Find Alex's friend Taylor, and then Taylor's friend Jordan.

Estimation: summarization and temporal changes
How does Jordan's friend group change over the course of the year?
Topics

Tree Visualization

Graph Layout: Node-Link Diagrams

Alternatives

Tools
Tree Visualization
Tree Visualization

Indentation
Linear list, indentation encodes depth

Node-Link diagrams
Nodes connected by lines/curves

Enclosure diagrams
Represent hierarchy by enclosure

Layering
Relative position and alignment

Typically fast: O(n) or O(n log n), interactive layout
Indentation
**Indentation**

Places 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
Single-Focus (Accordion) List

Separate breadth & depth along 2D. Focus on a single path at a time.
What tasks are these good for?

Benefits:
Navigation + Browsing, Parent-Child Relationships

Disadvantages:
Estimation, Comparison, Network Overview
Node-Link Diagrams
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 (e.g., towards authority or generality).
Naïve Recursive Layout

Repeatedly divide space for subtrees by leaf count
Breadth of tree along one dimension
Depth along the other dimension
Naïve Recursive Layout

Repeatedly divide space for subtrees by leaf count
Breadth of tree along one dimension
Depth along the other dimension
Problems?
Naïve Recursive Layout

Repeatedly divide space for subtrees by leaf count
Breadth of tree along one dimension
Depth along the other dimension
Problem: exponential growth of breadth
Reingold & Tilford’s “Tidy” Layout

Goal: make smarter use of space, maximize density and symmetry.

Originally binary trees, extended by Walker to cover general case.

Corrected by Buchheim et al. to achieve a linear time algorithm.
Reingold-Tilford Layout

Design Considerations

- Clearly encode depth
- No edge crossings
- Draw isomophic subtrees identically (same shape)
- Preserve layout ordering and symmetry
- Compact, space-saving layout (don’t waste space)
Radial Tree Layout

Node-link diagram in polar co-ordinates.

Radius encodes depth, with root in the center.

Angular sectors assigned to subtrees (often with naïve recursive layout).

Reingold-Tilford method can also be applied here.
Analysis Tasks: Focus+Context
Visualizing Large Hierarchies

Indented Layout

Reingold-Tilford Layout
More Nodes, More Problems…

Scale
Tree breadth often grows exponentially
Even with tidy layout, quickly run out of space

Possible Solutions
Filtering
Focus+Context
Scrolling or Panning
Zooming
Aggregation
Perform tree layout in hyperbolic geometry, project the result onto the Euclidean plane.

Why? Like tree breadth, the hyperbolic plane expands exponentially!

Also computable in 3D, projected into a sphere.
Hyperbolic Layout
Degree-of-Interest Trees

Space-constrained, multi-focal tree layout
Degree-of-Interest Trees

Remove “low interest” nodes at a given depth level until all blocks on a level fit within bounds. Attempt to center child blocks beneath parents.
What tasks are supported/missing?
Indentation & Node-Link Diagrams

Encode structure in **2D space** (breadth/depth)

**Benefits**
Clearly depicts node relationships / structure
Structure-based or browsing tasks

**Problems**
Even with tidy layout, quickly run out of space

**Missing**
Attribute-based encodings
Enclosure
Enclosure Diagrams

Encode structure using **spatial enclosure**
Popularly known as **treemaps**

**Benefits**
Provides a single view of an entire tree
Easier to spot large/small nodes

**Problems**
Difficult to accurately read structure / depth
Nodes are represented as sized circles. 
Nesting shows parent-child relationships. 

Issues?
Inefficient use of space. 
Parent size misleading?
Treemaps

Hierarchy visualization that emphasizes values of nodes via area encoding. Partition 2D space such that leaf nodes have sizes proportional to data values. First layout algorithms proposed by Shneiderman et al. in 1990, with focus on showing file sizes on a hard drive.
Slice & Dice layout: Alternate horizontal / vertical partitions.
Squarified layout: Try to produce square (1:1) aspect ratios
Slice & Dice layout suffers from extreme aspect ratios. How might we do better?

Squarified layout: greedy optimization for objective of square rectangles. Slice/dice within siblings; alternate whenever ratio worsens.

Why Squares? [Bruls et al. ‘00]

Posited Benefits of 1:1 Aspect Ratios

1. Minimize perimeter, reducing border ink. *Mathematically true!*
2. Easier to select with a mouse cursor. *Validated by empirical research & Fitt’s Law!*
3. Similar aspect ratios are easier to compare. *Seems intuitive, but is this true?*
Comparison Error vs. Aspect Ratio

Study by Kong, Heer & Agrawala, InfoVis ‘10.
Comparison of squares has higher error!
“Squarify” works because it fails to meet its objective?
Why Squares? [Bruls et al. ’00]

Posited Benefits of 1:1 Aspect Ratios

1. Minimize perimeter, reducing border ink.  
   *Mathematically true!*

2. Easier to select with a mouse cursor.  
   *Validated by empirical research & Fitt’s Law!*

3. Similar aspect ratios are easier to compare.  
   *Seems intuitive, but is this true?*
Why Squares? [Bruls et al. ’00]

Posited Benefits of 1:1 Aspect Ratios

1. Minimize perimeter, reducing border ink. *Mathematically true!*

2. Easier to select with a mouse cursor. *Validated by empirical research & Fitt’s Law!*

3. Similar aspect ratios are easier to compare. *Extreme ratios & squares-only more inaccurate.*

*Balanced ratios better? Target golden ratio?*
Interactive Example...
Cascaded Treemaps [Lü & Fogarty ’08]

Uses 2.5D effect to emphasize hierarchy relations.
Layering
Layered Diagrams

Signify tree structure using:
- Layering
- Adjacency
- Alignment

Involves recursive sub-division of space.

Leaf nodes may be sized by value, parent size visualizes sum of descendant leaf values.
Icicle Trees: Cartesian Partition
“Sunburst” Trees: Polar Partition

2.00% of visits begin with this sequence of pages.
Layered Trees Useful Elsewhere...

<table>
<thead>
<tr>
<th></th>
<th>Coffee</th>
<th>Espresso</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Amaretto</td>
<td>Caffe Latte</td>
</tr>
<tr>
<td></td>
<td>Columbian</td>
<td>Caffe Mocha</td>
</tr>
<tr>
<td></td>
<td>Decaf Irish Cr.</td>
<td>Decaf Espresso</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Regular Espresso</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Region</th>
<th>Colorado</th>
<th>Illinois</th>
<th>Iowa</th>
<th>Missouri</th>
<th>Ohio</th>
<th>Wisconsin</th>
</tr>
</thead>
<tbody>
<tr>
<td>Central</td>
<td>-20K</td>
<td>0 K</td>
<td>20K</td>
<td>-20K</td>
<td>0 K</td>
<td>20K</td>
</tr>
<tr>
<td>East</td>
<td>-20K</td>
<td>0 K</td>
<td>20K</td>
<td>-20K</td>
<td>0 K</td>
<td>20K</td>
</tr>
<tr>
<td>South</td>
<td>-20K</td>
<td>0 K</td>
<td>20K</td>
<td>-20K</td>
<td>0 K</td>
<td>20K</td>
</tr>
<tr>
<td>West</td>
<td>-20K</td>
<td>0 K</td>
<td>20K</td>
<td>-20K</td>
<td>0 K</td>
<td>20K</td>
</tr>
</tbody>
</table>
Node-Link Graph Layout
Node-Link Graph Visualization

Nodes connected by lines/curves

Sugiyama-Style Layout - arranged by depth

Force-Directed Layout - physical simulation

Attribute-Driven Layout - arranged by value

Constraint-Based Layout - optimization

Arc Diagrams - aligned layout
Sugiyama-Style Layout
Evolution of the UNIX operating system
Hierarchical layering based on descent

Sugiyama-Style Layout

GraphViz package!
Produce Hierarchical Layouts

Sugiyama-style layout emphasizes hierarchy. However, cycles in the graph may mislead. Long edges can impede perception of proximity.
Force-Directed Layout
Zephoria

User ID: 21721
Friends: 0
Age: ??
Gender: Female
Status: Single
Location: San Francisco, CA
Hometown: Lancaster, PA
Occupation: researcher, social networks, identity, context

Interests: apophenia, observing people, culture, questioning power, reading, Buddhism, gossip, computer-mediated communication, social networks, technology, anthropology, stumping


Books: Authors: Erving Goffman, Stanley Milgram, Jeannette Winterson, Eric Schlosser, Leslie Fanti, Dorothy Allison, Italo Calvino, Hermann Hesse

TV Shows: ??

Movies: Koyaanisqatsi, Amelie, Walking Life, Tank Girl, The Matrix, Clockwork Orange, American Beauty, Fight Club, Boys Don't Cry

Member Since: ??
Last Login: 2003-10-21
Last Updated: 2003-10-21

About: Some know me as danah...

I'm a geek, an activist and an academic, fascinated by people and society. I see life as a very large playground and enjoy exploring its intricacies. I revel in life's chaos, while simultaneously providing my own insane element.

My musings: http://www.zephoria.org/thoughts/

Want to Meet: Someone who makes life's complexities seem simply
Interactive Example: Configurable Force Layout
Use the Force!

http://mbostock.github.io/d3/talk/20110921/
d3.force
7,922 nodes
11,881 edges

[Kai Chang]
Different forces can be composed to create an expressive space of custom layouts.

A **beeswarm plot** can be made by combining:

- Attractive **X** and **Y** forces to draw nodes of a certain category to a desired point
- **Collide** force to detect collision & remove overlap
Attribute-Driven Layout
How many **herbivores** have no **predators**?
How many **herbivores** have no **predators**?
How many **herbivores** have no **predators**?
Attribute-Driven Layout

Large node-link diagrams get messy!
Is there additional structure we can exploit?

Idea: Use data attributes to perform layout
For example, scatter plot based on node values

Attributes may be associated with nodes or edges or may be statistical properties of the graph.

Use dynamic queries / brushing to explore…
Attribute-Driven Layout

The “Skitter” Layout
Internet Connectivity
Radial Scatterplot

Angle = Longitude
Geography

Radius = Degree
# of connections
(a statistic of the nodes)
Drawing all edges is not particularly useful here...
Node layout determined by geographic location.
Adjacent edges shown on node selection.
PivotGraph [Wattenberg ’06]

Layout aggregate graphs using node attributes. Analogous to pivot tables and trellis display.
PivotGraph

Node and Link Diagram

PivotGraph Roll-up
Limitations of PivotGraph

Only 2 variables (no nesting as in Tableau)
Doesn’t support continuous variables
Multivariate edges?
Constraint-Based Layout
Constraint-Based Layout

Treat layout as an optimization problem
Define layout using an energy model along with constraints: equations the layout should obey.
Use optimization algorithms to solve

Position Constraints:
a must be to the left of b
d, c, and b must have the same x position
a, b, and e must have the same y position
Optimizing Aesthetic Constraints

Minimize edge crossings
Minimize area
Minimize line bends
Minimize line slopes
Maximize smallest angle between edges
Maximize symmetry

but, can’t do it all.

Optimizing these criteria is often NP-Hard, requiring approximations.
SetCoLa: High-Level Layout

(1) Define **sets** of nodes based on attributes.
(2) Apply **constraints** to set elements.

**Layout using SetCoLa:**

(1) on all nodes
   (i) position left of "rbound"
   (ii) position right of "lbound"
(2) partition type
   (iii) padding 18
(3) compose set from types
   (iv) order by type

[Hoffswell '18]
Arc Diagrams
Linear node layout, circular arcs show connections. Layout quality sensitive to node ordering!
For example, the picture above was built from the first line of a very simple piece: *Mary Had a Little Lamb*. Each arch connects two identical passages. To clarify the connection between the visualization and the song, in this diagram the score is displayed beneath the arches.

`The Shape of Song
[Wattenberg ‘01]`

This diagram visualizes the refrain from the folk song *Clementine*. As you would expect, the refrain consists of multiple repetitions of the same passage—and that is exactly what the diagram shows. The score isn't shown in this diagram since the notes would be too small to read.
Task Analysis
Node-Link Graph Visualization

Nodes connected by lines/curves

**Sugiyama-Style Layout** - arranged by depth

**Force-Directed Layout** - physical simulation

**Attribute-Driven Layout** - arranged by value

**Constraint-Based Layout** - optimization

**Arc Diagrams** - aligned layout
Node-Link Graph Visualization

Nodes connected by lines/curves

Sugiyama-Style Layout

**The Good:** Structured-based analysis of hierarchical relationships

**The Bad:** Browsing and path following due to long edges

Force-Directed Layout

Attribute-Driven Layout

Constraint-Based Layout

Arc Diagrams
Node-Link Graph Visualization

Nodes connected by lines/curves

Sugiyama-Style Layout
Force-Directed Layout
Attribute-Driven Layout
Constraint-Based Layout
Arc Diagrams
Node-Link Graph Visualization

Nodes connected by lines/curves

Sugiyama-Style Layout

Force-Directed Layout

The Good: Structured-based analysis of closely related elements
The Bad: Browsing and summarization of dense networks

Attribute-Driven Layout

Constraint-Based Layout

Arc Diagrams
Node-Link Graph Visualization

Nodes connected by lines/curves

Sugiyama-Style Layout
Force-Directed Layout
Attribute-Driven Layout
Constraint-Based Layout
Arc Diagrams
Node-Link Graph Visualization

Nodes connected by lines/curves

Sugiyama-Style Layout

Force-Directed Layout

Attribute-Driven Layout

The Good: Attribute-based analysis tasks

The Bad (Difficult): Designing layouts appropriately

Constraint-Based Layout

Arc Diagrams
Node-Link Graph Visualization

Nodes connected by lines/curves

Sugiyama-Style Layout
Force-Directed Layout
Attribute-Driven Layout
Constraint-Based Layout
Arc Diagrams
Node-Link Graph Visualization

Nodes connected by lines/curves

Sugiyama-Style Layout

Force-Directed Layout

Attribute-Driven Layout

Constraint-Based Layout

**The Good:** Graph layout based on structural/aesthetic properties

**The Bad (Difficult):** Selecting constraints appropriately

Arc Diagrams
Node-Link Graph Visualization

Nodes connected by lines/curves

Sugiyama-Style Layout
Force-Directed Layout
Attribute-Driven Layout
Constraint-Based Layout
Arc Diagrams
Node-Link Graph Visualization

Nodes connected by lines/curves

Sugiyama-Style Layout
Force-Directed Layout
Attribute-Driven Layout
Constraint-Based Layout
Arc Diagrams

The Good: Summarization and comparison of overall structure
The Bad: Order matters for node layout; Structure-based and path following
Limitations of Node-Link Layouts

Edge-crossings and occlusion! Poor scalability....
Hierarchical Edge Bundling
Hierarchical Edge Bundling

Bundle edges with varying amounts of tension
Low-level vs. high-level information
Flare Class Hierarchy & Dependency Graph
Matrix Diagrams
Adjacency Matrices
Tools
What tools should you use?

Vega: variety of out-of-the-box support
D3.js: flexible and customizable
Graphviz: DOT language for graph layout
Cytoscape: software platform with app extensions
Gephi: software platform with plugins
What tools should you use?

Vega: variety of out-of-the-box support

D3.js: flexible and customizable

Graphviz: DOT language for graph layout

Cytoscape: software platform with app extensions

Gephi: software platform with plugins
What tools should you use?

Vega: variety of out-of-the-box support
D3.js: flexible and customizable
Graphviz: DOT language for graph layout
Cytoscape: software platform with app extensions
Gephi: software platform with plugins
What tools should you use?

Vega: variety of out-of-the-box support

D3.js: flexible and customizable

Graphviz: DOT language for graph layout

Cytoscape: software platform with app extensions

Gephi: software platform with plugins
What tools should you use?

Vega: variety of out-of-the-box support
D3.js: flexible and customizable
Graphviz: DOT language for graph layout
Cytoscape: software platform with app extensions
Gephi: software platform with plugins
What tools should you use?

Vega: variety of out-of-the-box support

D3.js: flexible and customizable

Graphviz: DOT language for graph layout

Features

The Graphviz layout programs take descriptions of graphs in a simple text language, and make diagrams in useful formats, such as images and SVG for web pages; PDF or Postscript for inclusion in other documents; or display in an interactive graph browser. Graphviz has many useful features for concrete diagrams, such as options for colors, fonts, tabular node layouts, line styles, hyperlinks, and custom shapes.

Cytoscape: software platform with app extensions

Gephi: software platform with plugins
What tools should you use?

Vega: variety of out-of-the-box support
D3.js: flexible and customizable
Graphviz: DOT language for graph layout
Cytoscape: software platform with app extensions
Gephi: software platform with plugins
What tools should you use?

Vega: variety of out-of-the-box support

D3.js: flexible and customizable

Graphviz: DOT language for graph layout

Cytoscape: software platform with app extensions

Cytoscape is an open source software platform for visualizing complex networks and integrating these with any type of attribute data. A lot of Apps are available for various kinds of problem domains, including bioinformatics, social network analysis, and semantic web. Learn more...

Gephi: software platform with plugins
What tools should you use?

Vega: variety of out-of-the-box support
D3.js: flexible and customizable
Graphviz: DOT language for graph layout
Cytoscape: software platform with app extensions
Gephi: software platform with plugins

APLICATIONS
✓ Exploratory Data Analysis: intuition-oriented analysis by networks manipulations in real time.
✓ Link Analysis: revealing the underlying structures of associations between objects.
✓ 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.

TECHNOLOGY
✓ Ergonomic interface: no programming skills needed
✓ High-performance: built-in rendering engine.
✓ Native file formats: GDF (GUESS), GraphML (NodeXL), GML, NET (Pajek), GEXF and more.
✓ Customizable by plugins: layouts, metrics, data sources, manipulation tools, rendering presets and more.

Learn More >
What tools should you use?

Vega: variety of out-of-the-box support

D3.js: flexible and customizable

Graphviz: DOT language for graph layout

Cytoscape: software platform with app extensions

Gephi: software platform with plugins

APPLICATIONS

✓ Exploratory Data Analysis: intuition-oriented analysis by networks manipulations in real time.
✓ Link Analysis: revealing the underlying structures of associations between objects.
✓ 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.

TECHNOLOGY

✓ Ergonomic interface: no programming skills needed
✓ High-performance: built-in rendering engine.
✓ Native file formats: GDF (GUESS), GraphML ('NodeXL), GML, NET (Pajek), GEXF and more.
✓ Customizable by plugins: layouts, metrics, data sources, manipulation tools, rendering presets and more.

Learn More >
Summary: Hierarchies & Networks

**Tree Layout**
Indented / Node-Link / Enclosure / Layers
Focus+Context techniques for scale

**Graph Layout**
“Sugiyama” Layout
Force-Directed Layout
Attribute-Driven Layout
Constraint Layout
Arc Diagrams
Matrix Diagrams