Tree Drawing Methods

Modified from the original lecture by Kai (Kevin) Xu

Terminology

Graph \( G=(V,E) \)

- \( V \) is the vertex set
- \( E \) is the edge set. Each edge is a pair of vertices.

A path \((u_1, u_2, \ldots, u_m)\) is a sequence of vertices so that \((u_k, u_{k+1})\) is an edge for \( k=1,2,\ldots,m-1 \).

A cycle \((u_1, u_2, \ldots, u_m)\) is path such that \((u_m, u_1)\) is an edge.

Binary tree: rooted tree in which every node has at most two children.

Subtree rooted at \( v \) is the subgraph of all vertices that have \( v \) as their "ancestor".

Depth of a vertex \( v \) is the number of edges between \( v \) and the root.

Height of a vertex \( v \) is the maximum number of edges between \( v \) and a leaf.

Height of a tree: height of its root.

Ordered tree: rooted tree with an ordering for the children of every vertex.

Leftmost child: first child

Rightmost child: last child

Leftmost subtree: subtree rooted at leftmost child

Rightmost subtree: subtree rooted at rightmost child

Deciding the y-coordinate \( y_u \) of each vertex \( u \)

Top down:

- Choose \( y_u \) to be the negative of the depth of \( u \)

Bottom up:

- Choose \( y_u \) to be the height of \( u \)

How can we compute the x-coordinate?

Very simple Divide and Conquer method:

Properties of the very simple D&C drawing:

1. y-coordinate represents tree depth
2. Parent is centered above its children
3. Left child is on the left, right child is on the right
In-class exercise: draw this tree
1. Quentin is the Chief Executive Officer
2. Pierre (Tokyo Office) reports to Quentin
3. Prakesh (London Office) reports to Quentin
4. Pam (Sydney Office) reports to Quentin
5. Rowena (Manager of international products) reports to Pierre
6. Robert reports to Pierre
7. Ruiri reports to Pierre
8. Naimah reports to Rowena
9. Nore reports to Rowena
10. Liam reports to Robert
11. Leila reports to Robert
12. Kumiko reports to Ruiri
13. Kathy reports to Ruiri
14. Ken reports to Ruiri
15. Karl reports to Ruiri
16. Kiefer reports to Ruiri
17. Alice reports to Prakesh
18. Andrew reports to Prakesh
19. Marie reports to Alice
20. Min reports to Alice
21. Brian reports to Pam
22. Bonheo reports to Pam
23. Barnaby reports to Pam
24. Byongu reports to Pam

Reingold-Tilford “Tidier Tree” Algorithm

Almost simple Divide and conquer
1. If there is only one node, then draw it.
2. Else
   a) Recursively apply the algorithm to draw the left and right subtrees.
   b) Move the drawings of subtrees until their horizontal distance on some layer equals 2.
   c) Place the root vertically one level above and horizontally half way between its children. If there is only one child place the root at horizontal distance 1 from the child.

Reingold-Tilford “Tidier Tree” Algorithm

Compute the left and right contour of vertex $v$:
- scan the right contour of the left subtree ($T'$) and the left contour of the right subtree ($T''$)
- accumulate the displacements of the vertices on the left & right contour
- keep the max cumulative displacement at any depth

Reingold-Tilford “Tidier Tree” Algorithm

Properties of the drawing
1. $y$ coordinate represents tree depth
2. Parent is centered above its children
3. Left child is on the left, right child is on the right
   ... 
   Also: Tries to reduce width - i.e., give small area and high resolution

Reingold-Tilford “Tidier Tree” Algorithm

References:
- Also in The Book, G. DiBattista, P. Eades, R. Tamassia, I.G. Tollis, Graph Drawing, Prentice Hall, 1999

Reingold-Tilford “Tidier Tree” Algorithm

Time Complexity
- Linear, but why?
- It is necessary to travel down the contours of two subtrees $T'$ and $T''$ only as far as the height of the subtree of lesser height
- The time spent processing vertex $v$ in the post-order traversal is proportional to the minimum heights of $T'$ and $T''$
- The sum is no more than the number of vertices of the tree
- Can be visualized by connecting vertices with same depth
Hence, the algorithm runs in linear time

Reingold-Tilford “Tidier Tree” Algorithm

Note
- Local horizontal compaction at each conquer step does not always compute a drawing of minimal width
- Can be solved in polynomial time using linear programming
- NP-hard if vertices are restricted to a grid.
Generalization to non-binary trees

More difficult than it seems

- root is placed at the average x-coordinates of its children
- small imbalance problem:
  - The picture show the result when the algorithm works from left to right.

Reference:

Tree Drawing Algorithm

Radial Drawings

Reference: The Book

(pause for a demo)

Tree Drawing Algorithm

(pause for a demo)

Radial Drawing

A variation of layered drawing

- Root at the origin
- Layers are concentric circles centered at the origin
- Usually draw each subtree in an annulus wedge \( W \)

Wedge Angle

- Choose wedge angle to be proportional to the leaf number in the subtree
- Problem: edge intersecting with level circle

Wedge angle

To guarantee planarity, define convex subset \( F \) of the wedge.
The tangent to circle \( c_i \) through \( v \) meet circle \( c_{i+1} \) at \( a \) and \( b \)
The unbounded region \( F \) formed by the line segment \( ab \) and the rays from origin through \( a \) and \( b \) is convex
The final wedge angle is the lesser between the angle of \( F \) and angle proportional to number of leaves.
Time and Area

Time:
- Linear

Area:
- Polynomial
- Equal distance between circles
- Tree height: \(h\)
- Maximum child number: \(d_M\)
- Area:
  - \(O(h^2d_M^2)\)
  - First circle has perimeter as least \(d_M\) (minimum distance between two vertices is 1)
  - Its radius is \(O(d_M)\)
  - The radius of final circle is \(O(hd_M)\)

Radial Drawing

Used for free trees (tree without a root)
- Select a root minimize tree height
- Can be found in linear time using simple recursive leaf pruning algorithm
- One or two centers

Variations:
- choice of root,
- radii of the circles,
- how to determine the wedge angle

Tree Drawing Algorithm

HV-Drawing (binary trees only)

References
Peter Eades, Tao Lin, Xuemin Lin, Minimum Size h-v Drawings, Advanced Visual Interfaces 1992: 385-394
Peter Eades, Tao Lin, Xuemin Lin, Two Tree Drawing Conventions, Int. J. Comput. Geometry Appl. 3(2) 133-153 (1993)

HV-Drawing – Binary Tree

HV-drawing of a binary tree \(T\):
- straight-line grid drawing such that for each vertex \(u\), a child of \(u\) is either
  - horizontally aligned with and to the right of \(u\), or vertically aligned with and below \(u\)
  - the bounding rectangles of the subtrees of \(u\) do not intersect

Planar, straight-line, orthogonal, and downward

Divide & Conquer Method

Divide: recursively construct hv-drawings for the left & right subtrees
Conquer: perform either
- a horizontal combination or
- a vertical combination
The height & width are each at most \(n-1\)

Right-Heavy-HV-Tree-Drawing

1. Recursively construct drawing of the left & right subtrees
2. Using only horizontal combination, place the subtree with the largest number of vertices to the right of the other one.

height of the drawing is at most \(\log n\)
Right-Heavy-HV-Tree-Drawing

HV-drawing (downward, planar, grid, straight-line and orthogonal)
Width is at most
\[ n - 1 \]
If the tree is balanced, then the height is at most
\[ \log_2 n \]
- The larger subtree is always placed to the right
- The size of parent subtree is at least twice the size of vertical child subtree
\[ \Rightarrow \text{area } O(n\log n) \]

Area-Aspect Ratio

Right-Heavy-HV-Tree-Draw
- Good area bound, but bad aspect ratio
Better aspect ratio:
- use both vertical and horizontal combinations
- Alternating the combination
  - Odd level: horizontal, even level: vertical
  - \[ O(n) \] area and constant aspect ratio

Optimization and Extension

It is possible to construct an HV-drawing of a binary tree that is optimal with respect to area or perimeter in \( O(n^2) \) time.

- Use dynamic programming approach

More Tree Visualization

Some examples of different tree visualization methods.

Indented Layout

Places all items along vertically spaced rows
Uses indentation to show parent-child relationships
Example: Windows explorer
Problems:
- Only showing part of the tree
- Bad aspect ratio (not space efficient)
But still the most popular one!

Dendrogram

Essentially a layered drawing
- with bent orthogonal edges
Layering is bottom-up:
- All the leaves are on the same layer
Now commonly used in bioinformatics
Balloon trees

- A variation of radial layout
- Children are drawn in a circle centered at their parents.
- Effective in showing the tree structure
  - At the cost of node detail

Hyperbolic Tree

Simulates the distortion effect of a fisheye lens

- Enlarge the focus and shrink the context

3D hyperbolic tree:

- Projecting a graph on a sphere produces a similar distortion effect
- This example also uses balloon tree drawing.

3D tree visualization - Cone tree

Cone trees are a 3D extension of the 2D layered tree drawing method.

- Parent at the tip of a cone, and its children spaced equally on the bottom circle of the cone
- Either horizontal or vertical

The extension to 3D does not necessarily mean more information can be displayed

- Occlusion problem
- Needs interaction

Other 3D tree visualizations

3D poly-plane tree visualization

- Put subtrees on planes
- Arrange these planes in 3D to reduce occlusion
- In this example, layered drawing is used within each plane

3D layered tree

- Only one cone
- Layers are the parallel circles on the surface
- Example: color indicate the layer

Space-filling methods - Treemap

Treemaps use containment to show the hierarchy.

- They partition space recursively according to the size of subtrees
- It is space-efficient compared to node-link diagram
- It is effective in showing the leaf nodes; on the other size, it is difficult to see the non-leaf nodes

Variations of treemap

Cushioned treemap

- Use shading to help identify the levels in a treemap

Voronoi treemap

- Similar idea but uses voronoi diagram as partition
- The space does not have to be rectangle.
**Space-filling tree layout**

- Try to use as much screen space as possible.
- Layout a tree according to the recursive partition of the screen space.
  - The area allocated to a subtree is proportional to its size.
  - A bigger example: 55000 nodes
    - Use all the screen space
    - Not very effective on showing the tree structure

**Other space filling methods - Icicle Trees**

- Edges implied by adjacency and spatial relationship.

**Information slice and Sunburst Diagrams**

- Information slice
  - also a space-filling visualization method.
  - Radial version of icicle trees.
  - Node size is proportional to the angle swept by a node.

- Sunburst
  - With extra focus+context
  - Details are shown outside or inside the ring

**TreeViewer**

- Visualizes trees in a form that closely resembles botanical trees
  - The root is the tree stem
  - Non-leave nodes are branches
  - Leave nodes are “bulbs” at the end of branches
  - Example: Unix home directory.