Algorithms for graph visualization

Divide and Conquer - Tree Layouts - Part 2
Feedback from the questionary

What have I learned in this lecture?

- Basic/simple/easy/cool algorithm for tree visualization in linear time
- Things that seem to be easy are very difficult
- Time complexity
- Separation into drawing conventions and aesthetics
- The algorithm for drawing a binary tree by dynamic planning?
- That a visual proof is legit
- Goals of the lecture (1!)
Feedback from the questionary

What was not clear?

- Time complexity
- Why boundary update is $O(1)$?
- Why called level based layout?
- Pseudocode?
- Running example
- Details in the data structure
- Boundaries?
- Example before the time complexity not clear
Feedback from the questionary

What was not clear?
- Time complexity
- Why boundary update is O(1)?
- Why called level based layout?
- Pseudocode?
- Running example
- Details in the data structure
- Boundaries?
- Example before the time complexity not clear

Can I work out the details myself?
- Yes
- Fundamentals missing

2 - 3
Feedback from the questionary

What would I like to learn about the tree layouts?

- Applications
- More layouts and more algorithms
- **Algorithm for non binary tree**
- Area/width optimization (K. J. Supowit, E. M. Reingold The complexity of drawing trees nicely [SR83])
- What kind of trees force the afterward optimization of the width?
- **Empirical findings**
- More efficient algorithm than $O(n)$?
- Is it easy to figure out which is the best parent node? (do you mean root node?) (best for what?)
- How the tree is constructed in the first place?
Overview

- H(horizontal) V(vertical) tree layout algorithm
- Radial tree layout algorithm
- Other visualization styles
Applications

Cons cell diagram in LISP.

*Cons*(constructs) are memory objects which hold two values or pointers to values.

![Cons cell diagram in LISP](http://gajon.org/)

**Figure 3:** Diagram of cons cells of the simple tree.

Discuss with your neighbour(s) and then share

2+3 min

4 - 1
Applications

Cons cell diagram in LISP.

Cons(constructs) are memory objects which hold two values or pointers to values.

![Cons cell diagram in LISP](http://gajon.org/)

**Figure 3**: Diagram of cons cells of the simple tree.

Discuss with your neighbour(s) and then share

- What are the drawing conventions and aesthetics?

**2+3 min**
HV-Layout

Drawing Conventions:
- Children are vertically and horizontally aligned with the root
- The bounding boxes of the children do not intersect

Drawing Aesthetics:
- Height, width, area
HV-Layout

Divide & Conquer Approach:
HV-Layout

Induction base: ●

Induction step: combine layouts

horizontal combination

vertical combination
Right-Heavy HV-Layout

**Right-Heavy approach:**
- At every induction step apply horizontal combination
- Place the larger subtree to the right
Right-Heavy HV-Layout

**Right-Heavy approach:**
- At every induction step apply horizontal combination
- Place the larger subtree to the right

**Lemma**

Let $T$ be a binary tree. The height of the drawing constructed by Right-Heavy approach is at most $\log n$. 
Right-Heavy HV-Layout

Right-Heavy approach:
- At every induction step apply horizontal combination
- Place the larger subtree to the right

Lemma
Let $T$ be a binary tree. The height of the drawing constructed by Right-Heavy approach is at most $\log n$.

Proof:
- Each vertical edge has length 1
- Let $w$ be the lowest node in the drawing
- Let $P$ be a path from $w$ to the root of $T$
- For every edge $(u, v)$ in $P$: $|T(v)| > 2|T(u)|$
- $\Rightarrow P$ contains at most $\log n$ edges
Right-Heavy HV-Layout

- At every induction step apply horizontal combination
- Place the larger subtree to the right

Discuss with your neighbour(s) and then share

- What are the implementational details of the algorithm?
- How to compute the coordinates? Can we do it in $O(n)$ time?
Right-Heavy HV-Layout

**Theorem**

Let $T$ be a binary tree with $n$ vertices. The Right-Heavy algorithm constructs in $O(n)$ time a drawing $\Gamma$ of $T$ such that:
Theorem

Let $T$ be a binary tree with $n$ vertices. The Right-Heavy algorithm constructs in $O(n)$ time a drawing $\Gamma$ of $T$ such that:

- $\Gamma$ is HV-drawing (planar, orthogonal)
Theorem

Let $T$ be a binary tree with $n$ vertices. The Right-Heavy algorithm constructs in $O(n)$ time a drawing $\Gamma$ of $T$ such that:

- $\Gamma$ is HV-drawing (planar, orthogonal)
- The width of $\Gamma$ is at most

Take a minute to think about the width of the layout

1 min
Theorem

Let $T$ be a binary tree with $n$ vertices. The Right-Heavy algorithm constructs in $O(n)$ time a drawing $\Gamma$ of $T$ such that:

- $\Gamma$ is HV-drawing (planar, orthogonal)
- The width of $\Gamma$ is at most $n-1$
Right-Heavy HV-Layout

Theorem

Let $T$ be a binary tree with $n$ vertices. The Right-Heavy algorithm constructs in $O(n)$ time a drawing $\Gamma$ of $T$ such that:

- $\Gamma$ is HV-drawing (planar, orthogonal)
- The width of $\Gamma$ is at most $n-1$
- The height is at most $\log n$
Right-Heavy HV-Layout

**Theorem**

Let $T$ be a binary tree with $n$ vertices. The Right-Heavy algorithm constructs in $O(n)$ time a drawing $\Gamma$ of $T$ such that:

- $\Gamma$ is HV-drawing (planar, orthogonal)
- The width of $\Gamma$ is at most $n-1$
- The height is at most $\log n$
- The area is $O(n \log n)$
Right-Heavy HV-Layout

**Theorem**

Let $T$ be a binary tree with $n$ vertices. The Right-Heavy algorithm constructs in $O(n)$ time a drawing $\Gamma$ of $T$ such that:

- $\Gamma$ is HV-drawing (planar, orthogonal)
- The width of $\Gamma$ is at most $n-1$
- The height is at most $\log n$
- The area is $O(n \log n)$
- Simply and axially isomorphic subtrees have congruent drawings, up to translation
### Theorem

Let $T$ be a binary tree with $n$ vertices. The Right-Heavy algorithm constructs in $O(n)$ time a drawing $\Gamma$ of $T$ such that:

- $\Gamma$ is HV-drawing (planar, orthogonal)
- The width of $\Gamma$ is at most $n-1$
- The height is at most $\log n$
- The area is $O(n \log n)$
- Simply and axially isomorphic subtrees have congruent drawings, up to translation

### General rooted tree:

```
               largest subtree
           ____         ____
          |          |          |
          v          v          v
```

10 - 8
Bad news We can not minimize the area by using divide & conquer approach
HV-Layout

Bad news We can not minimize the area by using divide & conquer approach
Good news We can compute minimum area using Dynamic Programming
HV-Layout

Bad news We can not minimize the area by using divide & conquer approach
Good news We can compute minimum area using Dynamic Programming

HV-Layout for Trees

- Book Di Battista et al: Chapter 3.1.4
- Skript: page 86

11 - 3
Applications

Radial layout

An unrooted phylogenetic tree for myosin, a superfamily of proteins. "A myosin family tree" Journal of Cell Science
Applications

An unrooted phylogenetic tree for myosin, a superfamily of proteins. "myosin family tree“ Journal of Cell Science
Applications

Flare Visualization Toolkit code structure by Heer, Bostock and Ogievetsky, 2010

Greek Myth Family by Ribecca, 2011
Radial Layout

Drawing Conventions:
- Vertices lie on circular layers according to their depth
- Drawing is planar

Drawing Aesthetics:
- Distribution of the vertices
Radial Layout

Drawing Conventions:
- Vertices lie on circular layers according to their depth
- Drawing is planar

Drawing Aesthetics:
- Distribution of the vertices

Take a minute to think about a possible algorithm to optimize the distribution of the vertices

1 min
Radial Layout

Example: Angle corresponding to the subtree rooted at $u$: $\tau_u = \frac{\ell(u)}{\ell(v) - 1}$
Radial Layout

Example: Angle corresponding to the subtree rooted at $u$: $\tau_u = \frac{\ell(u)}{\ell(v)-1}$

$\ell(u)$ $v$

$u$

Example:

11

9 7 5 3

1 1 1 1 1 1 1
Radial Layout

Example: Angle corresponding to the subtree rooted at $u$: $\tau_u = \frac{\ell(u)}{\ell(v)-1}$

Example diagram with angle calculation.
Radial Layout

Example: Angle corresponding to the subtree rooted at $u$: $\tau_u = \frac{\ell(u)}{\ell(v) - 1}$
Radial Layout

**Example:** Angle corresponding to the subtree rooted at $u$: $\tau_u = \frac{\ell(u)}{\ell(v)-1}$

$\ell(u)$

$v$

$\tau_u = \frac{\ell(u)}{\ell(v)-1}$

$9 \cdot \frac{1}{8}$

$\frac{1}{10}$

$16 - 5$
Example: 

Angle corresponding to the subtree rooted at $u$: $\tau_u = \frac{\ell(u)}{\ell(v) - 1}$

\[ \ell(u) \]
\[ u \]
\[ v \]

\[ \ell(u) \]
\[ u \]
\[ v \]

\[ \ell(u) \]
\[ u \]
\[ v \]

\[ \frac{9}{10} \cdot \frac{1}{8} \]
\[ \frac{1}{10} \]

16 - 6
Radial Layout

**Example:**

- Angle corresponding to the subtree rooted at \( u \):
  \[
  \tau_u = \frac{\ell(u)}{\ell(v) - 1}
  \]
Radial Layout

Example: Angle corresponding to the subtree rooted at $u$: $\tau_u = \frac{\ell(u)}{\ell(v) - 1}$
Radial Layout

How to avoid crossings:

17 - 1
Radial Layout

How to avoid crossings:
Radial Layout

How to avoid crossings:
Radial Layout

How to avoid crossings:
Radial Layout

How to avoid crossings:
Radial Layout

How to avoid crossings:

- $\tau_u$ - angle of the wedge corresponding to vertex $u$
- $\rho_i$ - radius of layer $i$
- $\ell(v)$ - number of nodes in the subtree rooted at $v$
- $\cos\left(\frac{\tau_u}{2}\right) = \frac{\rho_i}{\rho_{i+1}}$
Radial Layout

How to avoid crossings:

- $\tau_u$ - angle of the wedge corresponding to vertex $u$
- $\rho_i$ - radius of layer $i$
- $\ell(v)$ - number of nodes in the subtree rooted at $v$
- $\cos \frac{\tau_u}{2} = \frac{\rho_i}{\rho_{i+1}}$
Radial Layout

How to avoid crossings:

- \( \tau_u \) - angle of the wedge corresponding to vertex \( u \)
- \( \rho_i \) - radius of layer \( i \)
- \( \ell(v) \)-number of nodes in the subtree rooted at \( v \)
- \( \cos \frac{\tau_u}{2} = \frac{\rho_i}{\rho_i + 1} \)

\[
\tau_u = \min\left\{ \frac{\ell(u)}{\ell(v) - 1}, 2 \arccos \frac{\rho_i}{\rho_i + 1} \right\}
\]
(correction)
Radial Layout

How to avoid crossings:

- $\tau_u$ - angle of the wedge corresponding to vertex $u$
- $\rho_i$ - radius of layer $i$
- $\ell(v)$-number of nodes in the subtree rooted at $v$
- $\cos\frac{\tau_u}{2} = \frac{\rho_i}{\rho_i+1}$

$$\tau_u = \min\left\{\frac{\ell(u)}{\ell(v)-1}, 2 \arccos \frac{\rho_i}{\rho_i+1}\right\}$$ (correction)

Alternatively use number of leaves in the subtree to subdivide the angles [book Di Battista et al.]
Radial Layout

Discuss with your neighbour(s) and then share

- Why the produced drawing is planar?

- \( \ell(v) \)-number of nodes in the subtree rooted at \( v \)

- \( \cos \frac{\tau_u}{2} = \frac{\rho_i}{\rho_{i+1}} \)

- \( \tau_u = \min\{\frac{\ell(u)}{\ell(v)} - 1, 2 \arccos \frac{\rho_i}{\rho_{i+1}}\} \) (correction)

- Alternatively use number of leaves in the subtree to subdivide the angles [book Di Battista et al.]
Radial Layout

Theorem

Let $T$ be a rooted tree with $n$ vertices. The radial algorithm constructs in $O(n)$ time a drawing $\Gamma$ of $T$ such that:

- $\Gamma$ is planar
- Each vertex lies on the radial layer equal to its height
- The area of the drawing is at most $O(h^2d_M^2)$, $h$-height, $d_M$-max number of children

Assuming that the radii of consecutive layers differ by the same number and the distance between the vertices on the layer is at least one
Radial Layout

Theorem

Let $T$ be a rooted tree with $n$ vertices. The radial algorithm constructs in $O(n)$ time a drawing $\Gamma$ of $T$ such that:

- $\Gamma$ is planar
- Each vertex lies on the radial layer equal to its height
- The area of the drawing is at most $O(h^2d_M^2)$, $h$-height, $d_M$-max number of children

Assuming that the radii of consecutive layers differ by the same number and the distance between the vertices on the layer is at least one.
Radial Layout

Theorem

Let $T$ be a rooted tree with $n$ vertices. The radial algorithm constructs in $O(n)$ time a drawing $\Gamma$ of $T$ such that:

- $\Gamma$ is planar
- Each vertex lies on the radial layer equal to its height
- The area of the drawing is at most $O(h^2 d_M^2)$, $h$-height, $d_M$-max number of children

Assuming that the radii of consecutive layers differ by the same number and the distance between the vertices on the layer is at least one

radius is at least $d_M$

radius is at least $h d_M$
Radial Layout for Trees

- Book Di Battista et al: Chapter 3.1.3
- Skript: Chapter 6.1.2
Other Visualization Styles

Writing Without Words: the project explores methods of visually-representing text and visualises the differences in writing styles when comparing different authors.
Other Visualization Styles

Writing Without Words: the project explores methods of visually-representing text and visualises the differences in writing styles when comparing different authors.

similar to Ballon layout
Other Visualization Styles

A phylogenetically organised display of data for all placental mammal species.

Fractal tree layout
for more applications and layouts...