Visualization of Graphs

Lecture 1b:
Drawing Trees and Series-Parallel Graphs

Part I:
Layered Drawings

Jonathan Klawitter
(Rooted) Trees

**Leaf:** Vertex of degree 1

**Rooted tree:** tree with designated root

**Ancestor:** Vertex on path to root

**Parent:** Neighbor on path to root

**Successor:** Vertex on path away from root

**Child:** Neighbor not on path to root

**Depth:** Length of path to root

**Height:** Maximum depth of a leaf

**Binary Tree:** At most two children per vertex (left / right child)

3 traversals:

- **preorder:** node – left – right
- **inorder:** left – node – right
- **postorder:** left – right – node
First Grid Layout of Binary Trees

1. Choose $y$-coordinates: $y(u) = \text{depth}(u)$

2. Choose $x$-coordinates:

- **preorder**
- **inorder**
- **postorder**
Decision tree for outcome prediction after traumatic brain injury

Source: Nature Reviews Neurology
Layered Drawings – Applications

Family tree of LOTR elves and half-elves
Layered Drawings – Drawing Style

- What are properties of the layout?
- What are the drawing conventions?
- What are aesthetics to optimize?

Drawing conventions
- Vertices lie on layers and have integer coordinates
- Parent centered above children
- Edges are straight-line segments
- Isomorphic subtrees have identical drawings

Drawing aesthetics
- Area
- Symmetries
Layered Drawings – Algorithm

**Input:** A binary tree $T$

**Output:** A layered drawing of $T$

**Base case:** A single vertex

**Divide:** Recursively apply the algorithm to draw the left and right subtrees

**Conquer:**
Layered Drawings – Algorithm

**Input:** A binary tree $T$

**Output:** A layered drawing of $T$

**Base case:** A single vertex

**Divide:** Recursively apply the algorithm to draw the left and right subtrees

**Conquer:**

- some agreed distance
- sometimes 3 apart for grid drawing!
Layered Drawings – Algorithm Details

Phase 1 – postorder traversal:
- For each vertex compute horizontal displacement of left and right child
- \( x\text{-offset}(v_l) = -\left\lfloor \frac{d_v}{2} \right\rfloor \), \( x\text{-offset}(v_r) = \left\lceil \frac{d_v}{2} \right\rceil \)
- At vertex \( u \) (below \( v \)) store left and right contour of sub-tree \( T(u) \)
- Contour is linked list of vertex coordinates/offsets
- Find \( d_v = \text{min. horiz. distance between } v_l \text{ and } v_r \)

Phase 2 – preorder traversal:
- Compute \( x \)- and \( y \)-coordinates
Layered Drawings – Algorithm Details

Phase 1 – postorder traversal:
- For each vertex compute horizontal displacement of left and right child
- $x$-offset($v_l$) = $-\lceil \frac{d_v}{2} \rceil$, $x$-offset($v_r$) = $\lceil \frac{d_v}{2} \rceil$
- At vertex $u$ (below $v$) store left and right contour of sub-tree $T(u)$
- Contour is linked list of vertex coordinates/offsets
- Find $d_v = \text{min. horiz. distance between } v_l \text{ and } v_r$

Phase 2 – preorder traversal:
- Compute $x$- and $y$-coordinates

Runtime?
- How often do we have to walk along a contour?

$\Rightarrow \mathcal{O}(n)$
Layered Drawings – Result

Theorem. [Reingold & Tilford '81]
Let $T$ be a binary tree with $n$ vertices. We can construct a drawing $\Gamma$ of $T$ in $O(n)$ time, such that:

- $\Gamma$ is planar, straight-line and strictly downward
- $\Gamma$ is layered: y-coordinate of vertex $v$ is $-\text{depth}(v)$
- Horizontal and Vertical distances are at least 1
- Each vertex is centred wrt its children
- Area of $\Gamma$ is in $O(n^2)$ – but not optimal!
- Simply isomorphic subtrees have congruent drawings, up to translation
- Axially isomorphic subtrees have congruent drawings, up to translation and reflection

NP-hard
Layered Drawings – Result

Theorem. [Reingold & Tilford '81]
Let $T$ be a binary tree with $n$ vertices. We can construct a drawing $\Gamma$ of $T$ in $O(n)$ time, such that:
- $\Gamma$ is planar, straight-line and strictly downward
- $\Gamma$ is layered: y-coordinate of vertex $v$ is $-\text{depth}(v)$
- Horizontal and Vertical distances are at least 1
- Each vertex is centred wrt its children
- Area of $\Gamma$ is in $O(n^2)$ – but not optimal!
- Simply isomorphic subtrees have congruent drawings, up to translation
- Axially isomorphic subtrees have congruent drawings, up to translation and reflection

NP-hard
Visualization of Graphs

Lecture 1b:
Drawing Trees and Series-Parallel Graphs

Part II:
HV-Drawings

Jonathan Klawitter
**HV-Drawings – Drawing Style**

### Applications

- Cons cell diagram in LISP
- *Cons* (constructs) are memory objects which hold two values or pointers to values

### Drawing conventions

- Children are vertically or horizontally aligned with their parent
- The bounding boxes of the subtrees of the children are disjoint
- Edges are strictly down- or rightwards

### Drawing aesthetics

- Height, width, area

---

*Source: after gajon.org/trees-linked-lists-common-lisp/*
HV-Drawings – Algorithm

**Input:** A binary tree $T$
**Output:** An HV-drawing of $T$

**Base case:**

**Divide:** Recursively apply the algorithm to draw the left and right subtrees

**Conquer:**

horizontal combination
HV-Drawings – Algorithm

**Input:** A binary tree $T$

**Output:** An HV-drawing of $T$

**Base case:** ∅

**Divide:** Recursively apply the algorithm to draw the left and right subtrees

**Conquer:** horizontal combination

vertical combination
Lemma. Let $T$ be a binary tree. The drawing constructed by the right-heavy approach has
- width at most $n - 1$ and
- height at most $\log n$.

Right-heavy approach

- Always apply horizontal combination
- Place the larger subtree to the right

Size of subtree := number of vertices

How to implement this in linear time?
HV-Drawings – Result

**Theorem.**
Let $T$ be a binary tree with $n$ vertices. The right-heavy algorithm constructs in $O(n)$ time a drawing $\Gamma$ of $T$ s.t.:

- $\Gamma$ is an HV-drawing (planar, orthogonal, strictly right-/downward)
- Width is at most $n - 1$
- Height is at most $\log n$
- Area is in $O(n \log n)$
- 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 $T$ s.t.:

- $\Gamma$ is an HV-drawing (planar, orthogonal, strictly right-/downward)
- Width is at most $n - 1$
- Height is at most $\log n$
- Area is in $O(n \log n)$
- Simply and axially isomorphic subtrees have congruent drawings up to translation

General rooted tree

Optimal area?
Not with divide & conquer approach, but can be computed with Dynamic Programming.
Visualization of Graphs

Lecture 1b:
Drawing Trees and Series-Parallel Graphs

Part III:
Radial Layouts

Jonathan Klawitter
Radial Layouts – Applications

Phylogenetic tree
by Colicelli, ScienceSignaling, 2004
Radial Layouts – Applications

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

Greek Myth Family by Ribecca, 2011
Radial Layouts – Drawing Style

**Drawing conventions**
- Vertices lie on circular layers according to their depth
- Drawing is planar

**Drawing aesthetics**
- Distribution of the vertices

How can an algorithm optimize the distribution of the vertices?
Radial Layouts – Algorithm Attempt

**Idea**

- Reserve area corresponding to size $\ell(u)$ of $T(u)$:
  
  $$\tau_u = \frac{\ell(u)}{\ell(v) - 1}$$

- Place $u$ in middle of area
Radial Layouts – How To Avoid Crossings
Radial Layouts – How To Avoid Crossings

- $\tau_u$ – angle of the wedge corresponding to vertex $u$
- $\ell(u)$ – number of nodes in the subtree rooted at $u$
- $\rho_i$ – radius of layer $i$
- $\cos \frac{\tau_u}{2} = \frac{\rho_i}{\rho_{i+1}}$
- $\tau_u = \min\{ \ell(u) - 1, 2 \arccos \frac{\rho_i}{\rho_{i+1}} \}$
- Alternative:
  - $\alpha_{\min} = \alpha_u - \arccos \frac{\rho_i}{\rho_{i+1}}$
  - $\alpha_{\max} = \alpha_u + \arccos \frac{\rho_i}{\rho_{i+1}}$
Radial Layouts – Pseudocode

\[ \text{RadialTreeLayout}(tree \ T, \ root \ r \in T, \ radii \ \rho_1 < \cdots < \rho_k) \]
\begin{align*}
\text{begin} \\
\quad \text{postorder}(r) \\
\quad \text{preorder}(r, 0, 0, 2\pi) \\
\text{return} \ (d_v, \alpha_v)_{v \in V(T)} \\
\quad \text{// vertex pos./polar coord.} \\
\end{align*}

\[ \text{postorder}(\text{vertex} \ v) \]
\begin{align*}
\quad \ell(v) &\leftarrow 1 \\
\quad \text{foreach child} \ w \text{ of} \ v \text{ do} \\
\quad &\text{postorder}(w) \\
\quad \ell(v) &\leftarrow \ell(v) + \ell(w) \\
\end{align*}

\[ \text{preorder}(\text{vertex} \ v, \ t, \alpha_{\text{min}}, \alpha_{\text{max}}) \]
\begin{align*}
\quad d_v &\leftarrow \rho_t \\
\quad \alpha_v &\leftarrow (\alpha_{\text{min}} + \alpha_{\text{max}})/2 \\
\quad \text{//output} \\
\quad \text{if} \ t > 0 \text{ then} \\
\quad &\alpha_{\text{min}} \leftarrow \max\{\alpha_{\text{min}}, \alpha_v - \arccos \frac{\rho_t}{\rho_{t+1}}\} \\
\quad &\alpha_{\text{max}} \leftarrow \min\{\alpha_{\text{max}}, \alpha_v + \arccos \frac{\rho_t}{\rho_{t+1}}\} \\
\quad \text{left} &\leftarrow \alpha_{\text{min}} \\
\quad \text{foreach child} \ w \text{ of} \ v \text{ do} \\
\quad &\text{right} \leftarrow \text{left} + \ell(w) \\
\quad &\ell(v) - 1 \cdot (\alpha_{\text{max}} - \alpha_{\text{min}}) \\
\quad \text{preorder}(w, t + 1, \text{left, right}) \\
\quad \text{left} &\leftarrow \text{right} \\
\end{align*}

Runtime? \( \mathcal{O}(n) \)

Correctness? \( \checkmark \)
Radial Layouts – Result

Theorem.
Let $T$ be a tree with $n$ vertices. The RadialTreeLayout algorithm constructs in $O(n)$ time a drawing $\Gamma$ of $T$ s.t.:
- $\Gamma$ is radial drawing
- Vertices lie on circle according to their depth
- Area quadratic in max degree times height of $T$
  (see [GD Ch. 3.1.3] if interested)
Other tree visualisation styles

Writing Without Words: The project explores methods to visualises the differences in writing styles of different authors.

Similar to balloon layout
Other tree visualisation styles

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

Fractal layout
Other tree visualisation styles

A language family tree – in pictures
Other tree visualisation styles
Visualization of Graphs

Lecture 1b:
Drawing Trees and Series-Parallel Graphs

Part IV:
Series-Parallel Graphs

Jonathan Klawitter
A graph $G$ is **series-parallel**, if
- it contains a single (directed) edge $(s, t)$, or
- it consists of two series-parallel graphs $G_1$, $G_2$ with sources $s_1$, $s_2$ and sinks $t_1$, $t_2$ that are combined using one of the following rules:

**Series composition**
- $s_1 = s_2$
- $t_1 = t_2$

**Parallel composition**
- $s_1 = s_2$

convince yourself that series-parallel graphs are planar
A decomposition tree of $G$ is a binary tree $T$ with nodes of three types: \textit{S}, \textit{P} and \textit{Q}-type.

- A \textit{Q}-node represents a single edge.
- An \textit{S}-node represents a series composition; its children $T_1$ and $T_2$ represent $G_1$ and $G_2$.
- A \textit{P}-node represents a parallel composition; its children $T_1$ and $T_2$ represent $G_1$ and $G_2$. 

![Diagram of a decomposition tree with nodes S, P, and Q and graphs G1 and G2.]
Series-Parallel Graphs – Decomposition Example
Computational complexity:
Linear time algorithms for $\mathcal{NP}$-hard problems
(e.g. Maximum Matching, MIS, Hamiltonian Completion)
Series-Parallel Graphs – Drawing Style

**Drawing conventions**
- Planarity
- Straight-line edges
- Upward

**Drawing aesthetics**
- Area
- Symmetry
Series-Parallel Graphs – Straight-Line Drawings

Divide & conquer algorithm using the decomposition tree

- Draw $G$ inside a right-angled isosceles bounding triangle $\Delta(G)$

Base case: Q-nodes  

Divide: Draw $G_1$ and $G_2$ first

Conquer:

- S-nodes / series composition
- P-nodes / parallel composition

Do you see any problem?
Series-Parallel Graphs – Straight-Line Drawings

Divide & conquer algorithm using the decomposition tree

- Draw $G$ inside a right-angled isosceles bounding triangle $\Delta(G)$

Base case: Q-nodes

Divide: Draw $G_1$ and $G_2$ first

Conquer:

- S-nodes / series composition
- P-nodes / parallel composition

Divide: Draw $G_1$ and $G_2$ first

Conquer:

- S-nodes / series composition
- P-nodes / parallel composition

change embedding!
Series-Parallel Graphs – Straight-Line Drawings

- What makes parallel composition possible without creating crossings?

- This condition is preserved during the induction step.

**Lemma.**
The drawing produced by the algorithm is planar.

Assume the following holds: the only vertex in angle($v$) is $s$.
Theorem.
Let $G$ be a series-parallel graph. Then $G$ (with variable embedding) admits a drawing $\Gamma$ that
- is upward planar and
- a straight-line drawing
- with area in $\mathcal{O}(n^2)$.
- Isomorphic components of $G$ have congruent drawings up to translation.
$\Gamma$ can be computed in $\mathcal{O}(n)$ time.
Theorem. [Bertolazzi et al. 94]

There exists a $2n$-vertex series-parallel graph $G_n$ such that any upward planar drawing of $G_n$ that respects the embedding requires $\Omega(4^n)$ area.

- $2 \cdot \text{Area}(G_n) < \text{Area}(\Pi)$
- $2 \cdot \text{Area}(\Pi) \leq \text{Area}(G_{n+1})$
- $4 \cdot \text{Area}(G_n) \leq \text{Area}(G_{n+1})$
Literature

- [GD Chapter 3] for divide and conquer methods for rooted trees and series-parallel graphs
- [Reingold, Tilford '81] “Tidier Drawings of Trees”
  original paper for level-based layout algo
- [Reingold, Supowit '83] “The complexity of drawing trees nicely”
  NP-hardness proof for area minimisation & LP
- treevis.net – compendium of drawing methods for trees