Algorithms for Graph Visualization

Summer Semester 2016
Lecture #4

Divide-and-Conquer Algorithms:
Trees and Series-Parallel Graphs

(based on slides from Martin Nöllenburg and Robert Görke, KIT)
Uses of Divide & Conquer

Well suited for inductively/recursively defined Graph Classes

Rooted Binary Trees:
1. draw the left subtree
2. draw the right subtree
3. combine together + draw root

Terminology

\[ \text{depth}(v): \text{distance from the root} \]

\[ \text{traversal} \]
- preorder
- inorder
- postorder
Overview

- balanced drawings of binary trees $O(nh)$
- radial drawings of trees $O(nh)$
- compact drawings of trees $O(n \log n)$
- upward drawings of series parallel graphs exp.
Algorithm of Reingold and Tilford ('81)

Trivial: If \( T = \{v\} \), draw \( v \) (e.g., as a small disk).

Divide: Run Alg. recursively on the left and right subtrees.

Conquer: shift the partial drawings to up to 2 units apart
place the root \( r \) one above and centrally btw. children.
Algorithm of Reingold and Tilford ('81)

Implementation in 2 Phases:

1. postorder (bottom-up):
   contours and x-offsets
   gather the predecessors

2. preorder (top-down):
   calculate absolute coordinates

Contour: linked list of vertices (-coordinates)
Algorithm of Reingold and Tilford ('81)

Phase 1:
1. compute $T_\ell(v)$ und $T_r(v)$
2. trace the right contour of $T_\ell(v)$ and left of $T_r(v)$
3. Find $d_v = \text{min. horiz. distance between } v_\ell \text{ und } v_r$
4. $x$-offset($v_\ell$) = $-\lceil d_v/2 \rceil$, $x$-offset($v_r$) = $\lceil d_v/2 \rceil$
5. Build left contour of $T_v$ from:
   - $v$,
   - left contour of $T_\ell(v)$,
   - left contour of any low hanging part of $T_r(v)$
6. Symmetrically for right contour.
Algorithm of Reingold and Tilford ('81)

Phase 1:
1. compute $T_\ell(v)$ und $T_r(v)$
2. trace the right contour of $T_\ell(v)$ and left of $T_r(v)$
3. Find $d_v = \text{min. horiz. distance between } v_\ell \text{ and } v_r$
4. $x\text{-offset}(v_\ell) = -\lceil d_v/2 \rceil$, $x\text{-offset}(v_r) = \lceil d_v/2 \rceil$
5. Build left contour of $T_v$ from:
   - $v,$
   - left contour of $T_\ell(v),$
   - left contour of any low hanging part of $T_r(v)$
6. Symmetrically for right contour.

Runtime? $\sum_v (1 + \min\{h_\ell(v), h_r(v)\}) = n + \sum_v \min\{\ldots\} \leq n + n$
Algorithm of Reingold und Tilford ('81)

Phase 2:

- Set $y$-coordinate $y(v) = -\text{depth}(v)$ for each vertex $v$.
- Set $x(w) := 0$ for the root $w$, then in preorder for $v \in V$:
  
  $x(v_l) := x(v) + x\text{-offset}(v_l)$ and
  
  $x(v_r) := x(v) + x\text{-offset}(v_r)$.

Runtime? $O(n)$
### Summary for Balanced Drawings of Binary Trees

**Theorem**  
[Reingold & Tilford '81]

For a binary tree with \( n \) vertices, in \( O(n) \) time we can produce a drawing \( \Gamma \) such that:

- \( \Gamma \) is layered, i.e., \( y \equiv - \text{depth} \),
- \( \Gamma \) is planar, straightline, and strictly downward,
- \( \Gamma \) matches the embedding (i.e., right children on the right),
- all vertices: horiz. & vert. distances \( \geq 1 \), and on the grid,
- the area is \( O(n^2) \),
- parent always centered above children.

**Min. width (but without the grid):** by LP!

**Min. width and on the grid: NP-hard!**  
[Supowit & Reingold '83]
Example of width variation

Output of the Algorithm:

Optimal Drawing:
2. Radial Drawings of Trees

- balanced drawings of binary trees $O(nh)$
- radial drawings of trees $O(nh)$
- compact drawings of trees $O(n \log n)$
- upward drawings of series parallel graphs $\exp.$
Example: Radial Tree Layouts
An Algorithm for Radial Layout?
Restricting to Smaller Sectors

\[ \cos \tau = \frac{\rho_i}{\rho_{i+1}} \]

\[ \alpha_{\text{min}} = \alpha_v - \arccos \left( \frac{\rho_i}{\rho_{i+1}} \right) \]

\[ \alpha_{\text{max}} = \alpha_v + \arccos \left( \frac{\rho_i}{\rho_{i+1}} \right) \]
Pseudocode for radial tree layout

RadialTreeLayout(tree $T$, root $r \in T$, radii $\rho_1 < \cdots < \rho_k$)

begin
  postorder($r$)
  preorder($r$, 0, 0, $2\pi$)
  return $(d_v, \alpha_v)_{v \in V(T)}$
  \{vertex pos./ polar coord.\}
end

postorder(vertex $v$)

$n_v \leftarrow 1$

foreach child $w$ von $v$ do
  postorder($w$)
  $n_v \leftarrow n_v + n_w$

preorder(vertex $v$, $t$, $\alpha_{\text{min}}$, $\alpha_{\text{max}}$)

$d_v \leftarrow \rho_t$

$\alpha_v \leftarrow (\alpha_{\text{min}} + \alpha_{\text{max}})/2$
\{ output \}

if $t > 0$ then
  $\alpha_{\text{min}} \leftarrow \max\{\alpha_{\text{min}}, \alpha_v - \arccos \frac{\rho_t}{\rho_{t+1}}\}$
  $\alpha_{\text{max}} \leftarrow \min\{\alpha_{\text{max}}, \alpha_v + \arccos \frac{\rho_t}{\rho_{t+1}}\}$

left $\leftarrow \alpha_{\text{min}}$

foreach child $w$ von $v$ do
  right $\leftarrow$ left $+ \frac{n_w}{n_v - 1} \cdot (\alpha_{\text{max}} - \alpha_{\text{min}})$
  preorder($w$, $t + 1$, left, right)
  left $\leftarrow$ right

size of the subtree $T(v)$

Runtime? $O(n)$.

Correctness? ✓
Overview

- balanced drawings of binary trees \( O(nh) \)
- radial drawings of trees \( O(nh) \)
- compact drawings of trees \( O(n \log n) \)
- upward drawings of series parallel graphs \( \exp \)
**Definition.**

An *hv-drawing* of a binary tree is a straight line drawing, so that for each vertex $v$:

- each child of $v$ is either directly right or directly below $v$.
- the smallest axis-parallel rectangle enclosing the subtrees of the children of $v$ are disjoint.

**hv-Drawings**

---

*horizontal combination*  
*vertical combination*
Algorithm \textit{RightHeavyHVTreeDraw}

\begin{itemize}
    \item Recursively construct drawings of the left and right subtrees from the root.
    \item Place the larger subtree on the right using the horizontal combination, and the smaller on the left
\end{itemize}

Size of a subtree := number of vertices

The drawing has width \( \leq n \), height \( \leq \log_2 n \).
Overview

- balanced drawings of binary trees $O(nh)$

- radial drawings of trees $O(nh)$

- compact drawings of trees $O(n \log n)$

- upward drawings of series parallel graphs $\exp.$
Series Parallel Graphs

- simple series parallel graph

- Induction: combining two series parallel graphs $G_1, G_2 \ldots$
  
  - ...series ...
  
  - ...or parallel.

$t_1 = s_2$

$s_1 = s_2$
Decomposition Tree for SP-graphs

Generalization: SPQR-Tree
SP-Graphs: applications

Flow Charts

Provides:

Linear time algorithms for NP-complete problems (e.g., Maximum Independent Set)

January 31, 2023
**Theorem** [Bertolazzi et al. '92]

There is a family \((G_n)_{n \in \mathbb{N}}\) of embedded SP-graphs where \(G_n\) has \(2^n\) vertices and every upward planar drawing of \(G_n\) requires \(\Omega(4^n)\) area.

**Proof:**

\[
a(G_{n+1}) \geq a(\Pi) + a(\Delta_1) + a(\Delta_2) \geq 2 \cdot a(\Pi) \geq 4 \cdot a(G_n)
\]