A Technique for Drawing Directed Graphs

by Emden R. Gansner, Eleftherios Koutsofios, Stephen C. North, Keim-Phong Vo

Jiwon Hahn
ECE238
May 13, 2003

Outline

- Introduction
- History
- DOT algorithm
  1. Ranking
  2. Ordering
  3. Positioning
  4. Drawing edges & labels
- Implementation (Graphviz)
- Conclusion

Drawing Graph

- Successful Techniques
  - Planarization
  - Layering (Hierarchical approach)
  - Physical Simulation

Drawing Hierarchical Graph

- Aesthetic criteria

  I. Expose hierarchical structure

  II. Avoid edge crossings and sharp bends

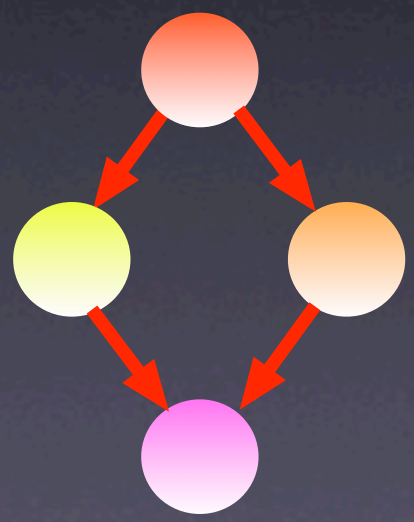
  III. Keep edges short

  IV. Favor symmetry and balance

Drawing Hierarchical Graph

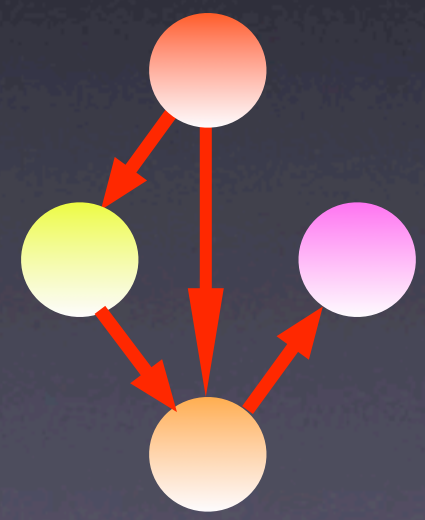
- Aesthetic criteria

  1. Expose hierarchical structure(edges in same direction)



(o)

VS.

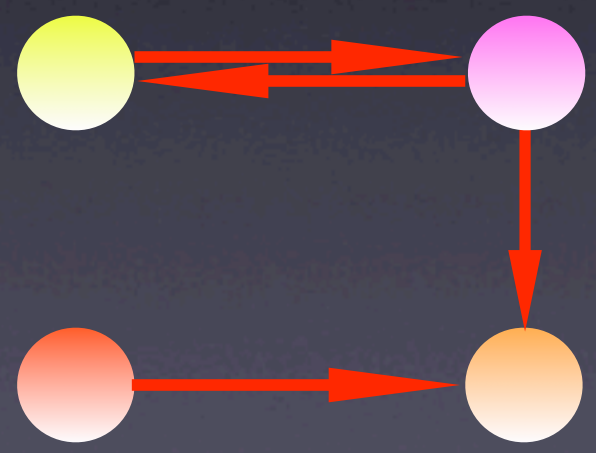


(x)

Drawing Hierarchical Graph

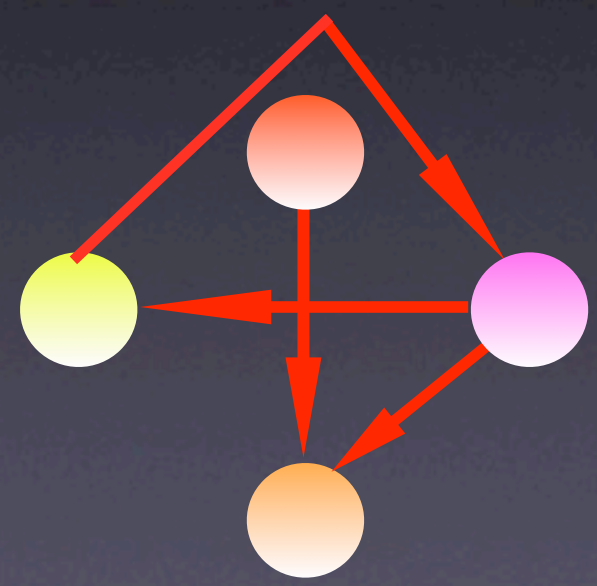
- Aesthetic criteria

  II. Avoid edge crossings and sharp bends



(o)

VS.

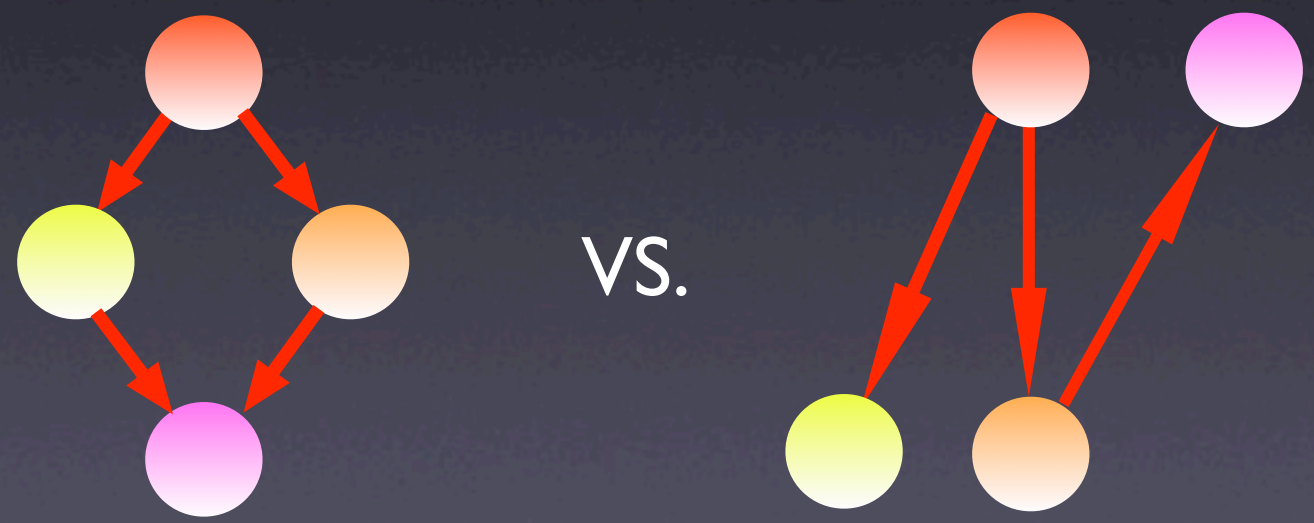


(x)

Drawing Hierarchical Graph

- Aesthetic criteria

  III. Keep edges short

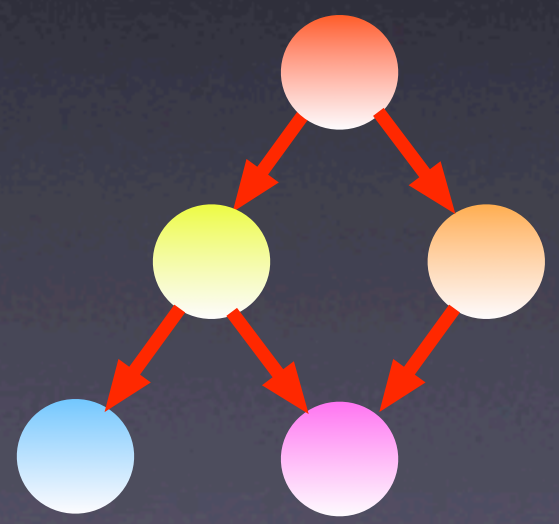


(o) (x)

Drawing Hierarchical Graph

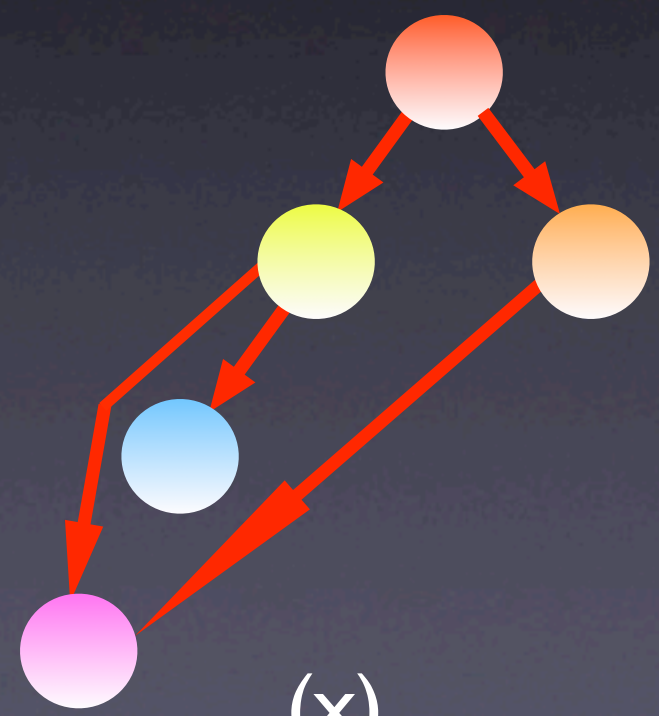
- Aesthetic criteria

  IV. Favor symmetry and balance



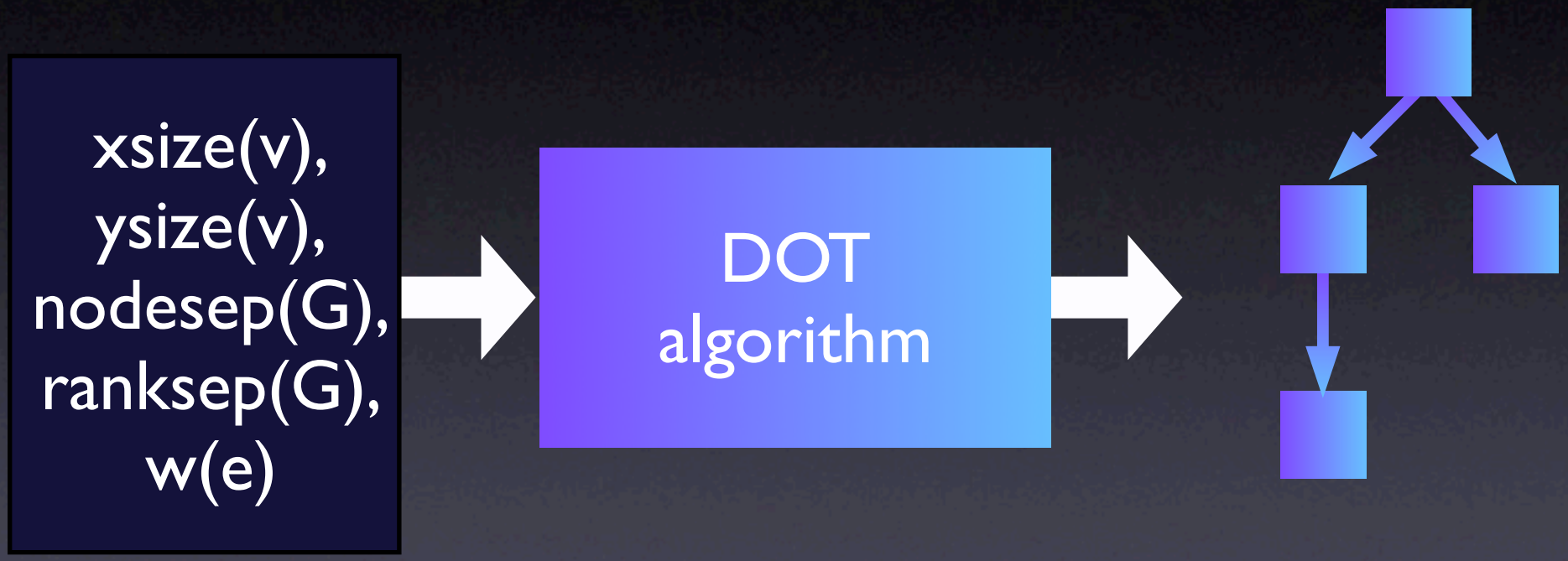
(o)

VS.



(x)

DOT algorithm



Impossible to optimize aesthetics I~IV simulataneouly!

Therefore, DOT relies on heuristics that run quickly and make good layouts in common cases

Outline

- Introduction
- History
- DOT algorithm
  1. Ranking
  2. Ordering
  3. Positioning
  4. Drawing edges & labels
- Implementation (Graphviz)
- Conclusion

History of Hierachical Visualization

- 1977: Warfield
- 1980: Carpano
- 1981: Sugiyama et al.
- 1988: DAG
- 1993: DOT → This work

Same basic approach
Greatly improved algorithms

History of Hierachical Visualization(cont.)

- 1993~ : DOT and DAG applied to various applications
  - Visualizing event streams in 2D
  - Drawing UML diagram
  - Drawing Automata Layouts
  - Visualizing Protein Interaction Pathways

Outline

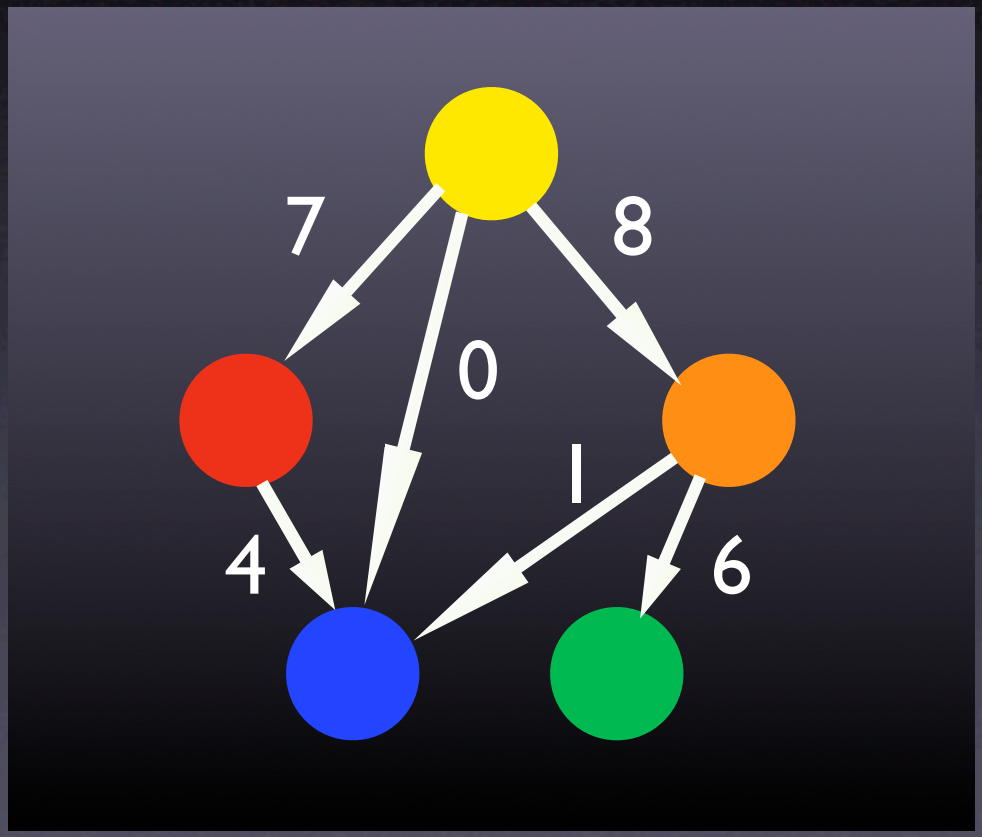
- Introduction
- History
- DOT algorithm
  1. Ranking
  2. Ordering
  3. Positioning
  4. Drawing edges & labels
- Implementation (Graphviz)
- Conclusion

DOT algorithm

```
1. procedure draw_graph()
2. begin
3.     rank();
4.     ordering();
5.     position();
6.     make_splines();
7. end
```

DOT algorithm(cont.)

1. Rank (Layering)
2. Ordering
3. Position
4. Make Splines +label

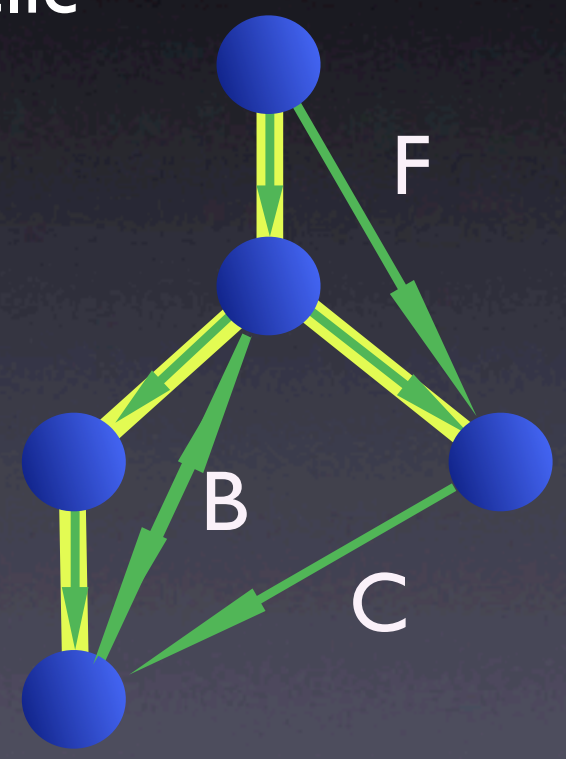


Outline

- Introduction
- History
- DOT algorithm
  1. Ranking
  2. Ordering
  3. Positioning
  4. Drawing edges & labels
- Implementation (Graphviz)
- Conclusion

Step 1. Ranking

- Preprocessing: Make the graph Acyclic
  - DFS heuristic
    - Among non-tree edges (Forward/Cross/Back edges), reverse the backward edges
    - Restore the reversed edges before drawing
    - More stable and informative drawings than other approaches



Step 1. Ranking (cont.)

- Assign each node to an integer rank
- Integer program for optimal ranking:
- $$min \sum_{(v,w)\ member\ E} \omega(v,w)(\lambda(w)-\lambda(v))$$
- $$subject\ to:\ \lambda(w) - \lambda(v) \geq \delta(v,w)\ \forall\ (v,w)\,memberE$$
- How to solve this?
  - ⭐ Network Simplex Algorithm (NSA)
    - not proven polynomial, but fast in practice
    - easy to program

Step 1. Ranking (cont.)

Definitions

- Simplex

  a popular linear programming algorithm that finds optimal solution by iteratively generating intermediate solution until optimal criterion is met

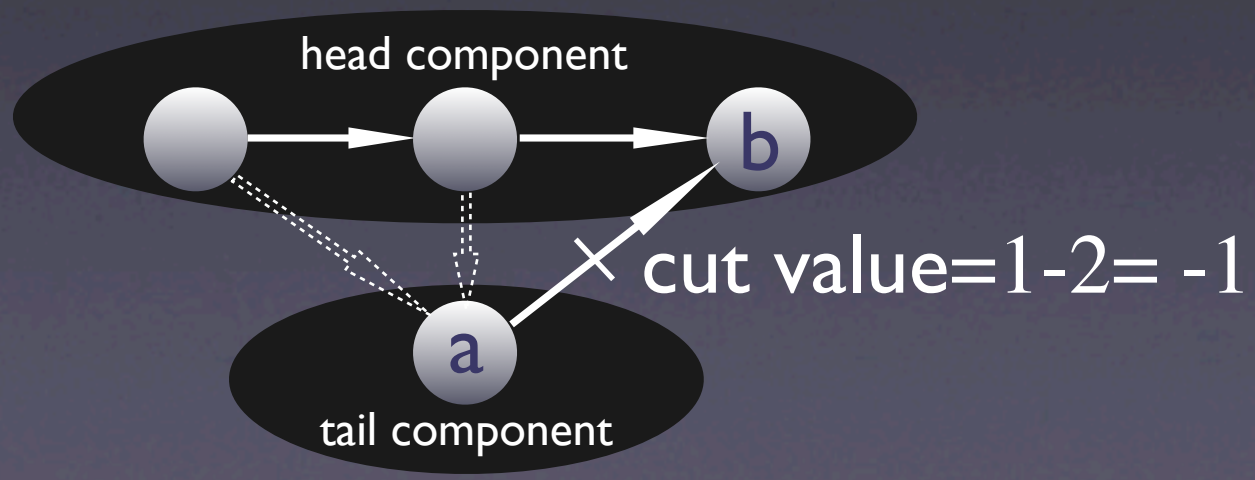
- Network Simplex

  variation of simplex that tries to make the # of iteration smaller by approaching in geometrical and graphical context rather than numerical one.

Step 1. Ranking (cont.)

Definitions

- Spanning tree
  a tree with |V|-1 edges that connects all the vertices of the Graph
- Feasible spanning tree
  a spanning tree that induces feasible ranking
- Cut value
  (sum of outgoing edges) - (sum of incoming edges) at the viewpoint of tail component



Step 1. Ranking (cont.)

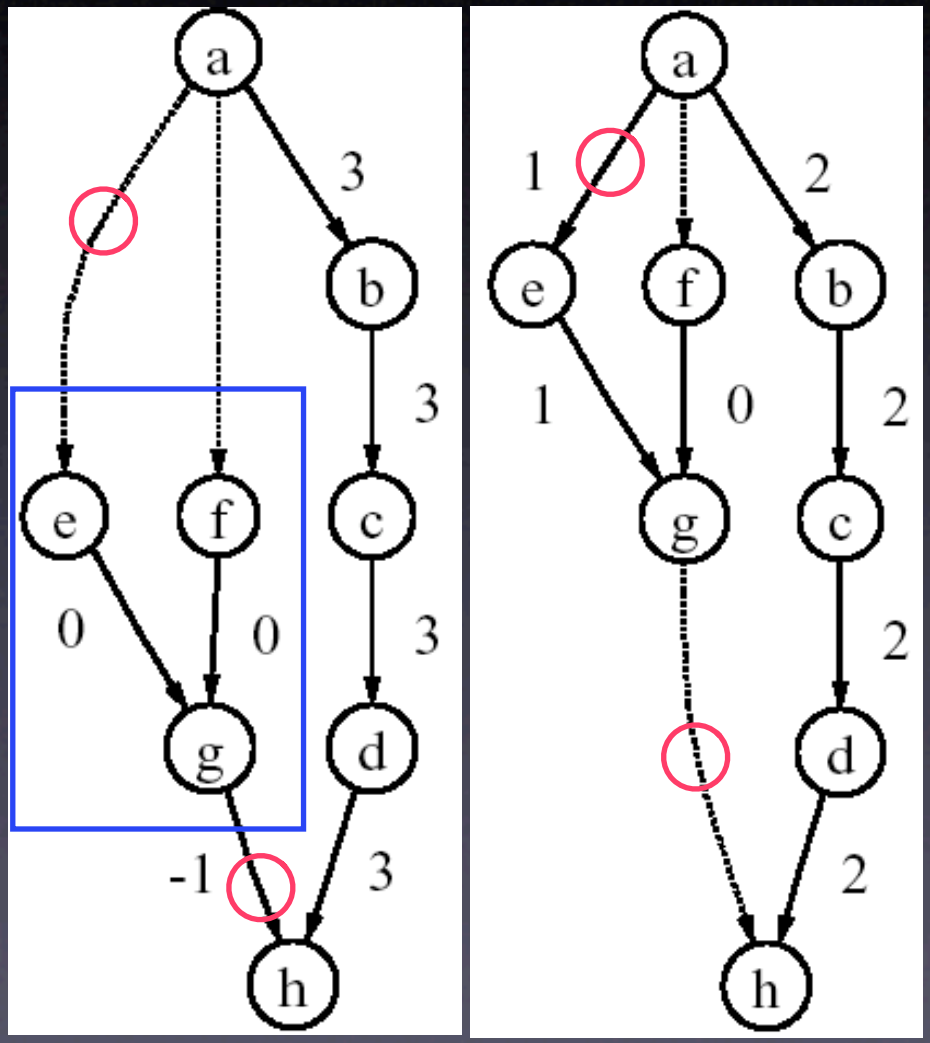
- Applying Network Simplex Algorithm to this problem:
  - Optimal solution is found when the cut values of each tree edge of a feasible spanning tree are non-negative
  - Typically, lengthening and replacing the tree edge w/ negative cut value helps

Step 1. Ranking (cont.)

Network Simplex Algorithm:

1. Construct a feasible spanning tree
2. While there exists a tree edge with negative cut, exchange it with non tree edge
3. Normalize (set least rank to zero)
4. Balance

Step 1. Ranking (cont.)



Example process of Network Simplex Algorithm

Step 1. Ranking (cont.)

Implementation issues:

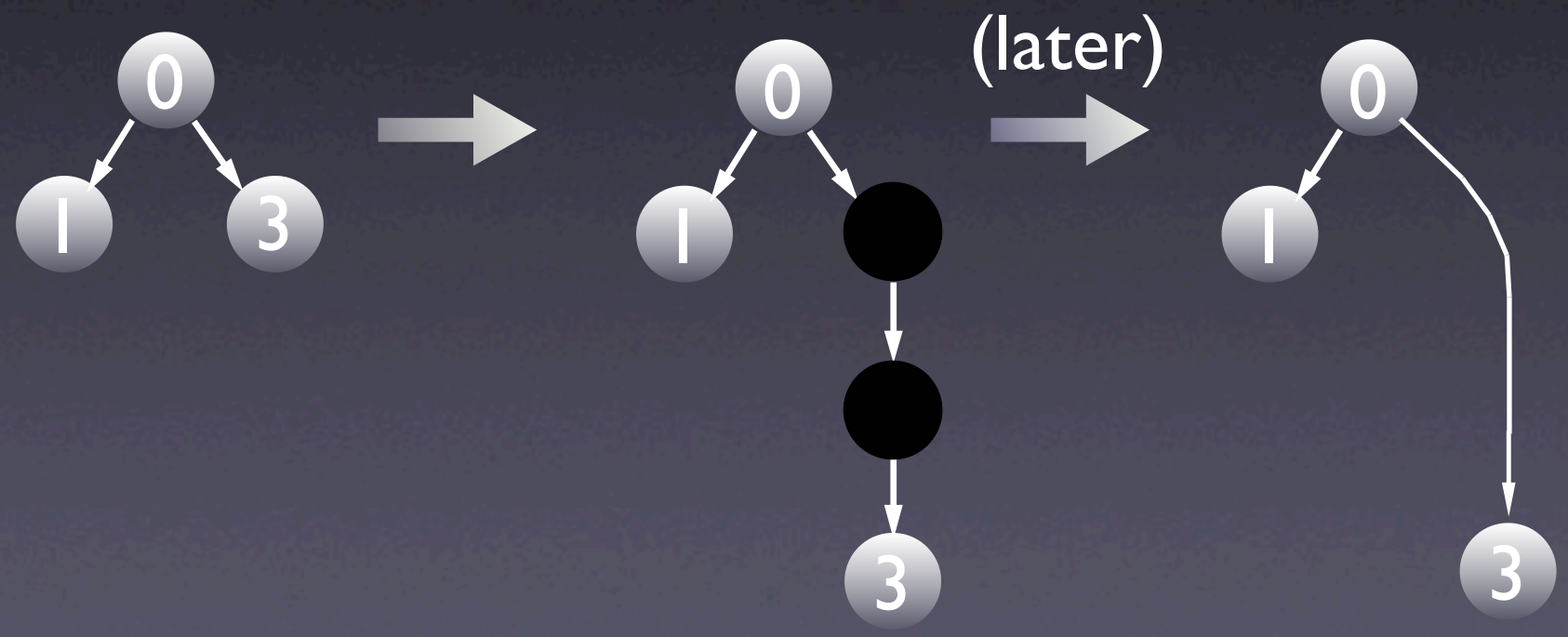
- Reduce the cost of cut value calculation
  - by using local edge info.
  - by indexing the tree according to postorder traversal
- Choose the best negative edge to replace
  - by searching cyclically

Outline

- Introduction
- History
- DOT algorithm
  1. Ranking
  2. Ordering
  3. Positioning
  4. Drawing edges & labels
- Implementation (Graphviz)
- Conclusion

Step 2. Ordering

- Preprocessing: replace the edges between nodes more than one rank apart with chains of virtual nodes



Step 2. Ordering(cont.)

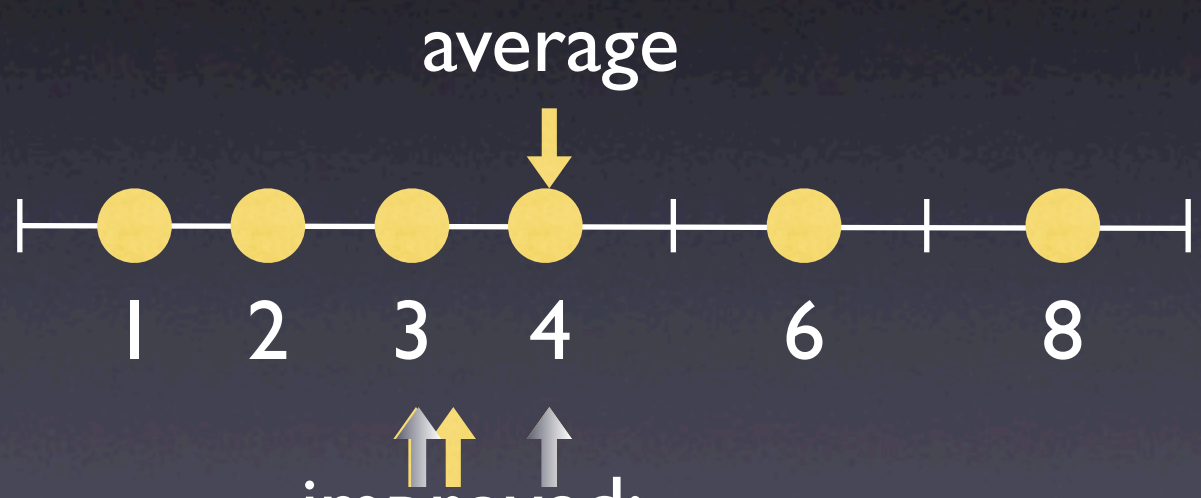
- Order vertices in the way that minimizes edge crossings (NP-complete)
- Edge crossing minimization heuristic:
  - suggested by Warfield in 1977
  - throughout iterations, improve ordering by iteratively assigning weight relative to previous rank and re-ordering based on weight

Step 2. Ordering(cont.)

- Vertex weighing methods:
  - barycenter function
    - average of adjacent nodes
  - median method
    - median of adjacent nodes
    - better performance
    - bounded number of edge crossing

Step 2. Ordering(cont.)

- Improved median by interpolation biased the closely packed vertices (at even nodes)



improved:

original median:

$$\frac{3\times(8-4)+4\times(3-1)}{(8-4)+(3-1)} = 3.3$$

choose among two

Step 2. Ordering(cont.)

- Ordering Algorithm

1. Initial ordering
2. While iteration:
   a. re-order the nodes based on improved median weighing method
   b. perform local transposition
   c. keep the best result

Outline

- Introduction
- History
- DOT algorithm
  1. Ranking
  2. Ordering
  3. Positioning
  4. Drawing edges & labels
- Implementation (Graphviz)
- Conclusion

Step 3. Positioning

- Set node coordinates (x, y) ➞ Y is easy
- Assigning X coordinates to vertices
  - Short, straight edges preferred
  - Captured in integer optimization problem

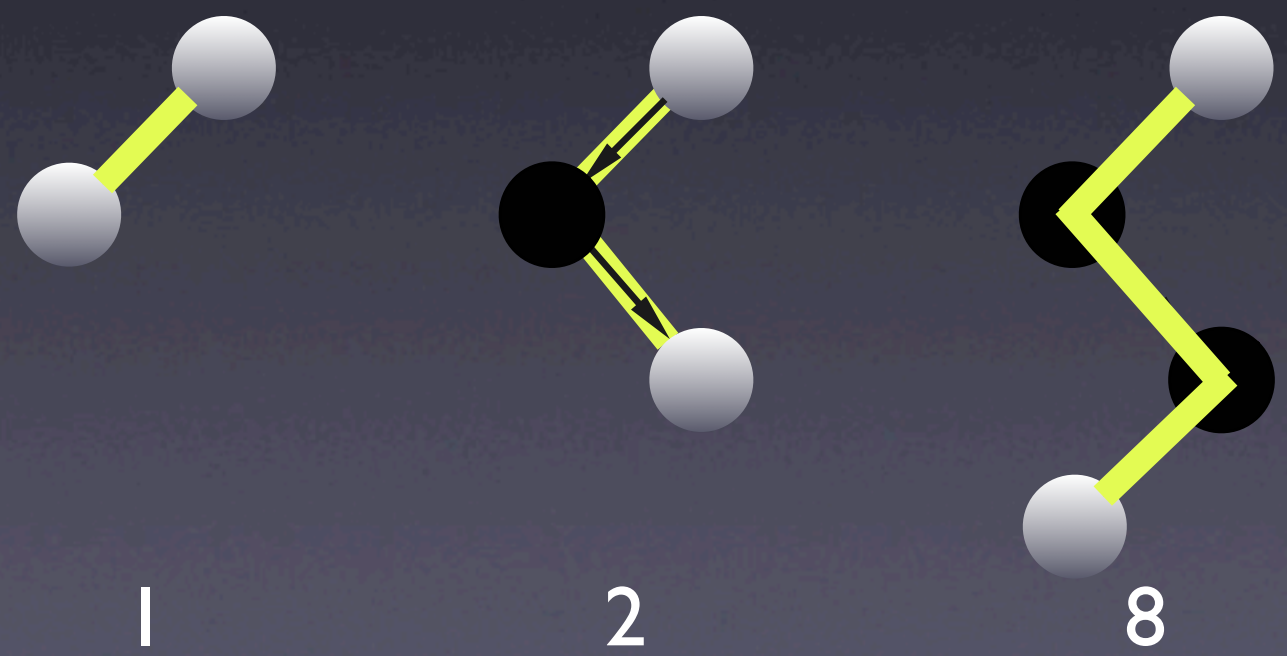
$$min \sum_{e=(v,w)} \Omega(e)\omega(e)\,|x_w - x_v|$$

$$subject\ to: x_b - x_a \geq \rho(a,b)$$

$$\rho(a,b) = \frac{xsize(a) + xsize(b)}{2} + nodesep(G)$$

Step 3. Positioning(cont.)

- Ω(e) : param. to reduce "spaghetti effect"
- Edges with virtual nodes get higher Ω(e)



Step 3. Positioning(cont.)

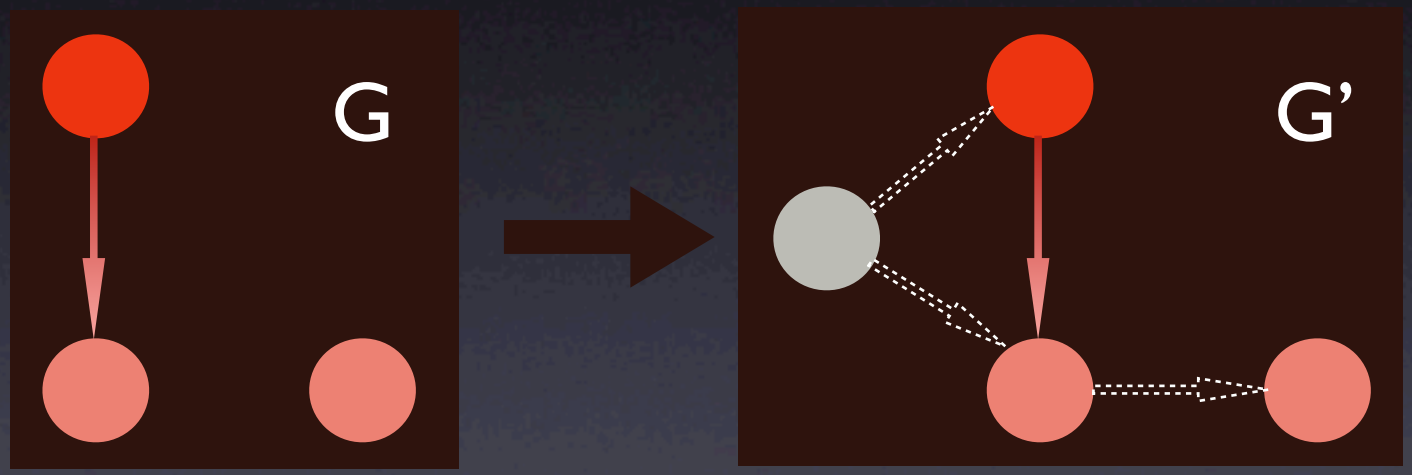
- Heuristic Approach
  - In each iter., a collection of heuristics applied
  - make good layout quickly
  - complicated to program
  - results are sometimes noticeably imperfect
  - not flexible (changes might result heuristics interfering with each other)

Step 3. Positioning(cont.)

- Optimal node placement
  - G → G' (auxiliary graph)
  - ★ Apply Network Simplex Algorithm to G'
    - repeatedly exchange negative-cut tree edge with non-tree edge
    - use x value instead of rank.

Step 3. Positioning(cont.)

- Constructing Auxiliary graph G'



- Encodes cost of original edges including Ω(e)
- Separates nodes in the same rank
- Enables specification of node ports
- Requires optimized implementation due to increased nodes and edges

Outline

- Introduction
- History
- DOT algorithm
  1. Ranking
  2. Ordering
  3. Positioning
  4. Drawing edges & labels
- Implementation (Graphviz)
- Conclusion

Step 4. Drawing edges

- Without overlapping other nodes & edges, draw the smoothest curve between two points
- Use splines rather than line segments
- Two parts:

1. Define region for edge layout
2. Compute the best spline within the region

Step 4. Drawing edges(cont.)

head & tail port boxes

virtual node boxes w/ extra space

inter-rank boxes

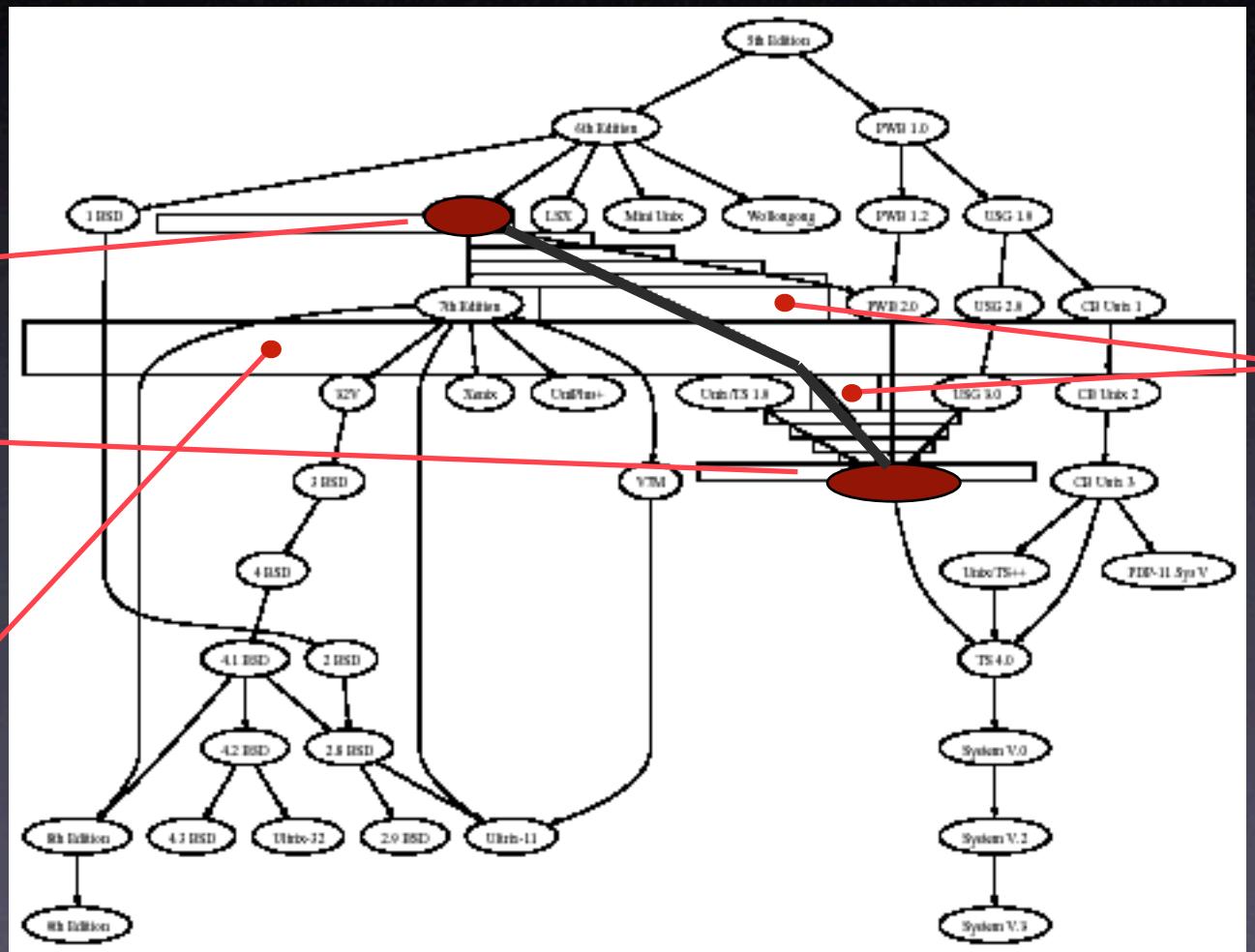


Figure 5-1. Region for a spline

(0.48 sec. user time Sun4-280)

Step 4. Drawing edges(cont.)

1. Finding the Region

- Setting the size of a box:
  - ignore horizontal virtual nodes connecting to edges within two ranks
- Long vertical section:
  - current region terminates and draw vertical line segment
- Common termination point:
  - subdivide the inter-rank space

Step 4. Drawing edges(cont.)

2. Computing Splines

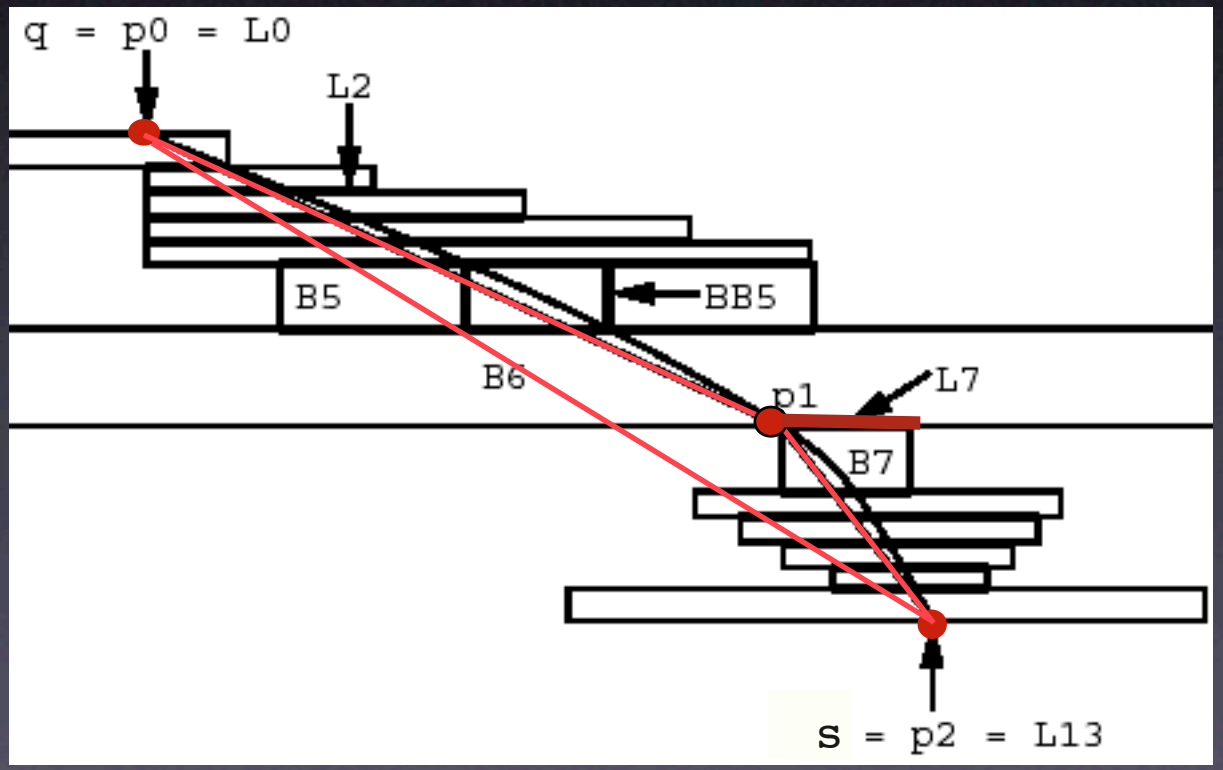
o. start with B[ ]

i. get L[ ]

ii. compute p[ ]

iii. compute s[ ]

iv. compute BB[ ]



Step 4. Drawing edges(cont.)

Edge Labels

- edge labels on inter-rank represented as virtual nodes
- double the ranks when virtual nodes are created → twice as expensive

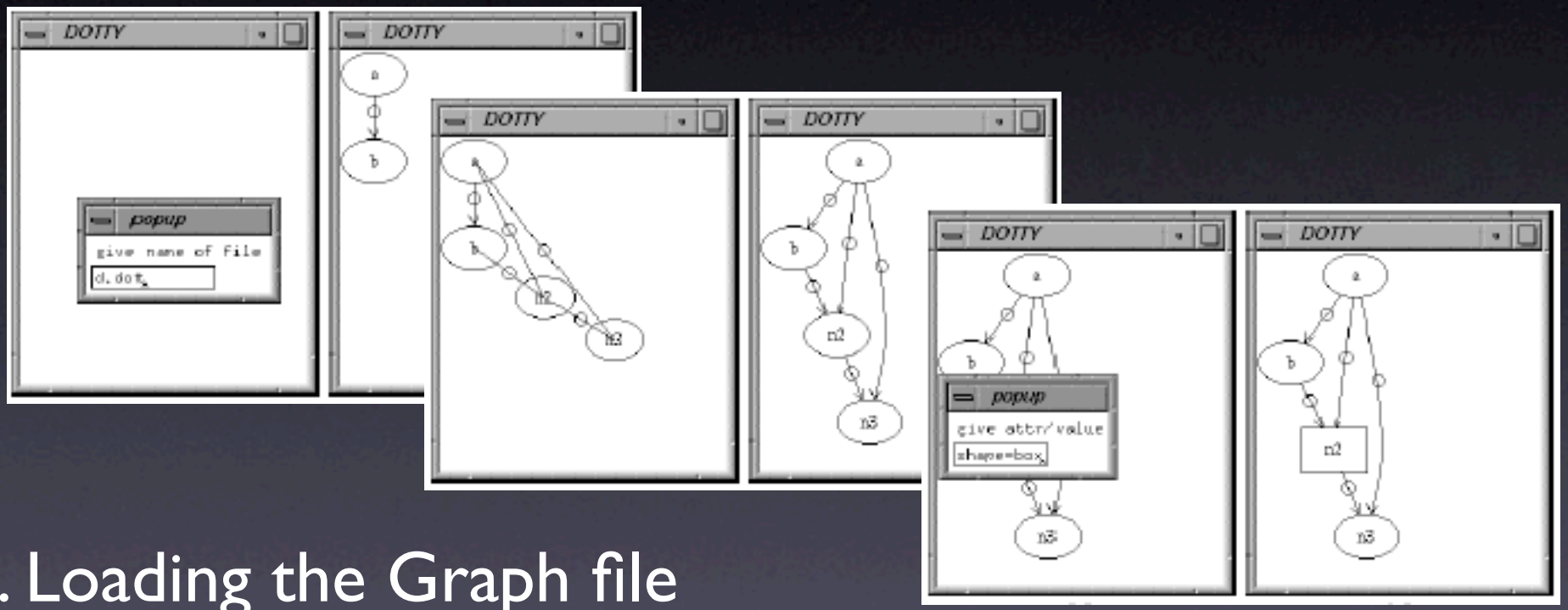
Outline

- Introduction
- History
- DOT algorithm
  1. Ranking
  2. Ordering
  3. Positioning
  4. Drawing edges & labels
- Implementation (Graphviz)
- Conclusion

Graphviz Package

- An open source program
- Available @ ATT homepage (http://www.research.att.com)
- Includes:
  - Algoritms: dot, neato, twopi
  - Programs: dot, dotty(tool-kit), grappa

Dotty: editing tool



a. Loading the Graph file

b. Inserting nodes and edges

c. Changing attributes

Drawing Graphs with *dot*

- *dot*: A Command Line program
- Input
  - .dot file written in DOT language

    describes graphs, nodes, edges
- Output
  - GIF, PNG, SVG, PostScript(→PDF)

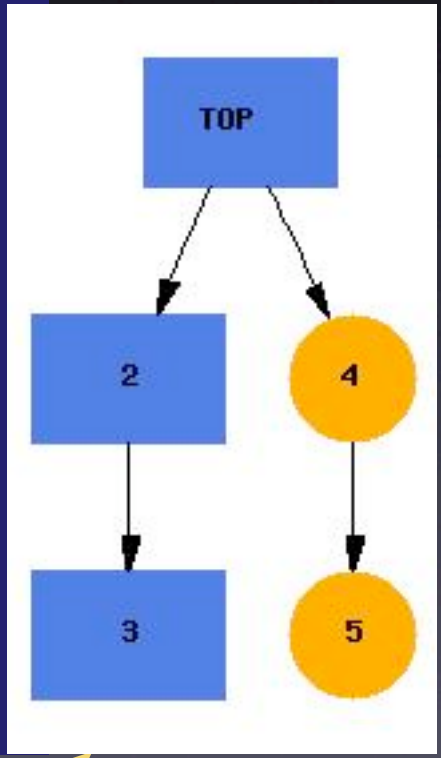
Drawing Graphs with *dot*

ex.dot

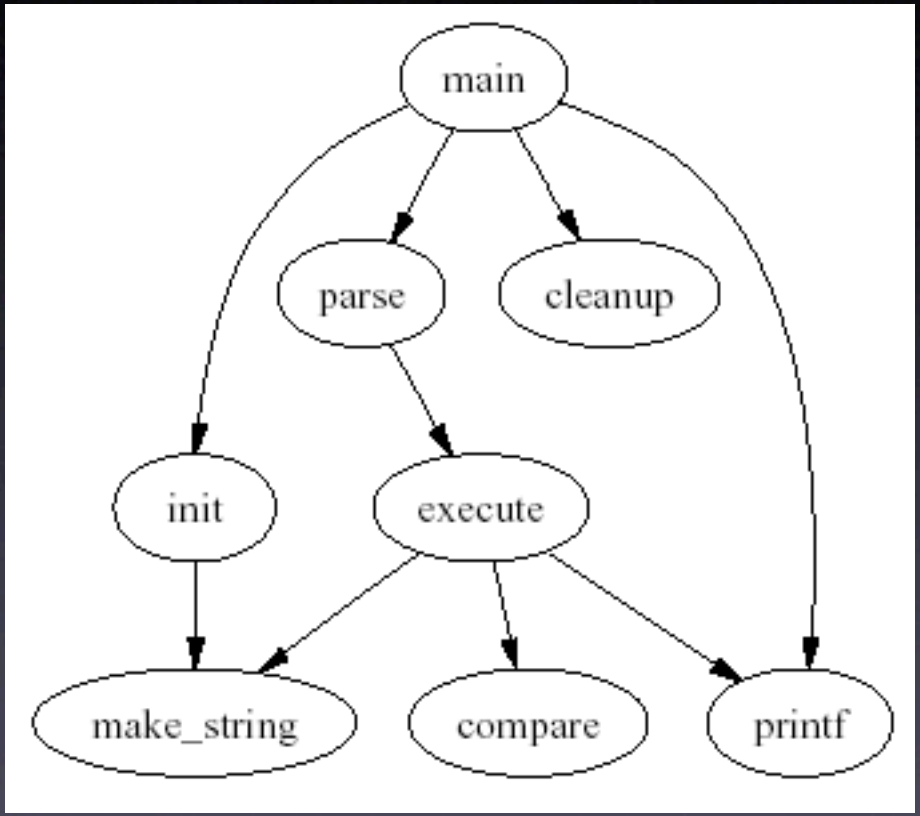
```
1.  digraph shells{
2.          size="4,5";
3.          node [fontsize=10, shape=box, color=royalblue];
4.          1 -> 2 -> 3;
5.          1 [label="TOP"];
6.          node [fontsize=10, shape=circle, color=orange];
7.          1 -> 4 -> 5;
8.          {rank=same; 2 4;}
9.          {rank=same; 3 5;}
10. }
```

```
% dot -Tjpg -o output.jpg ex.dot
```

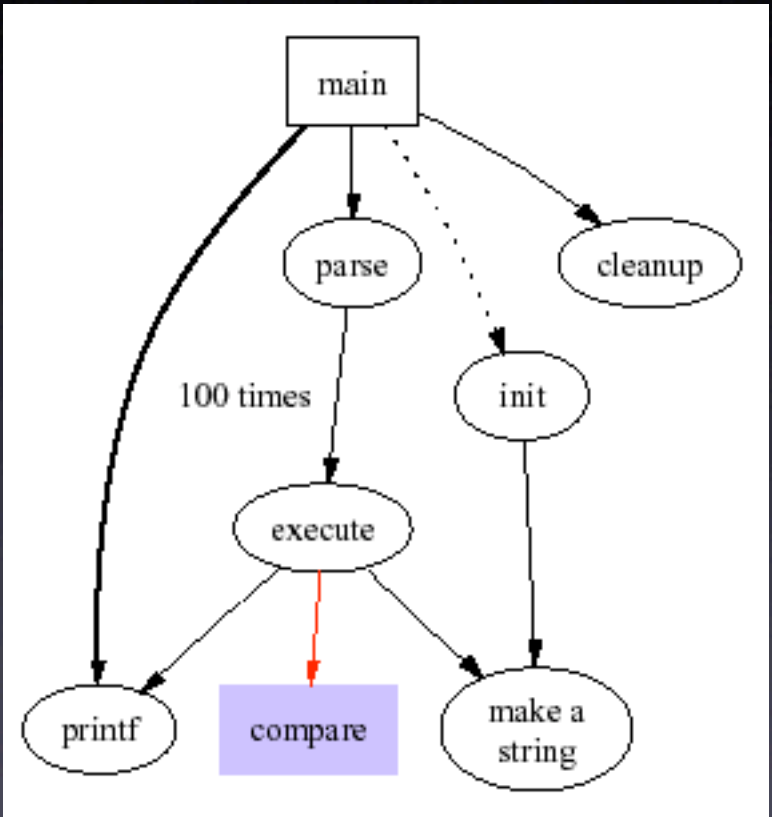
output.jpg



dot: Examples

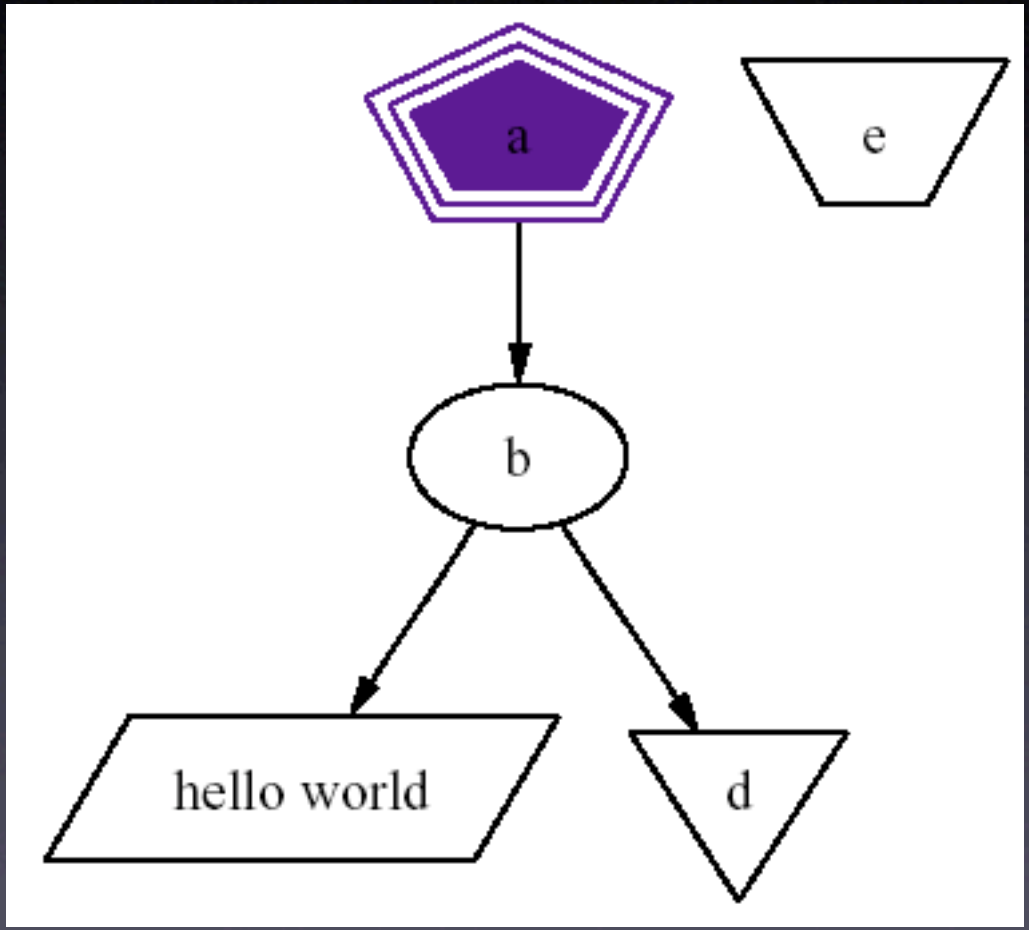


Simple graph



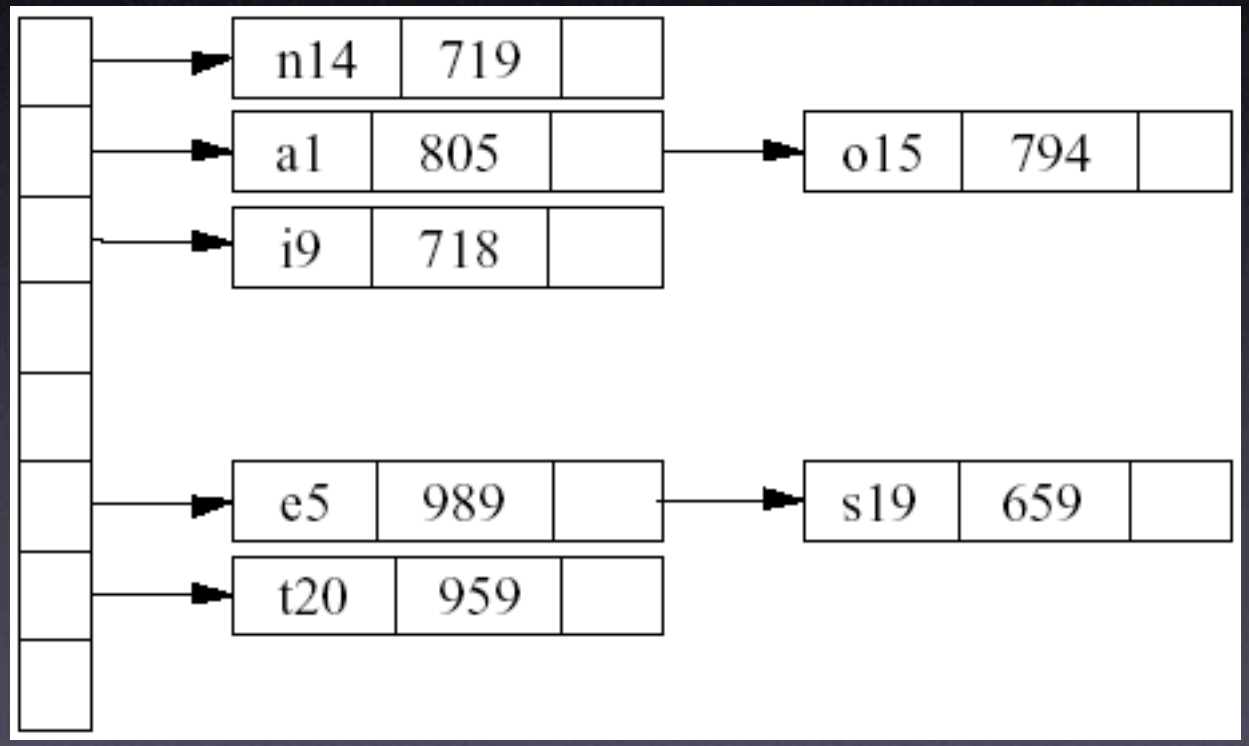
Fancy graph

dot: Examples (cont.)



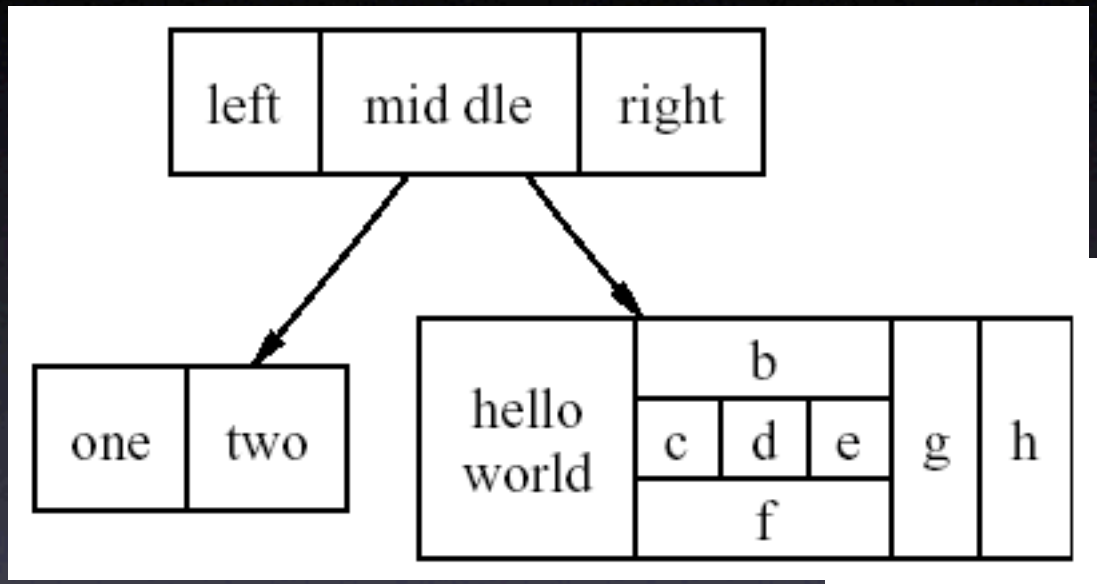
Polygonal Node Shapes

dot: Examples (cont.)

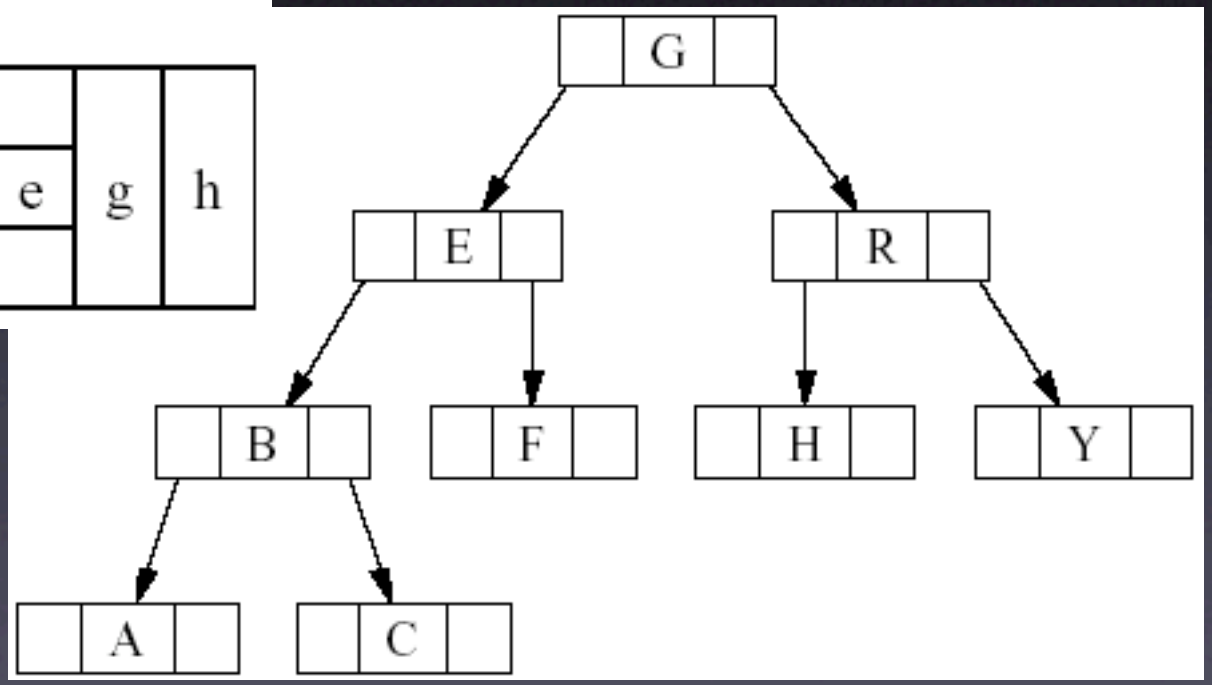


Hash Table

dot: Examples (cont.)

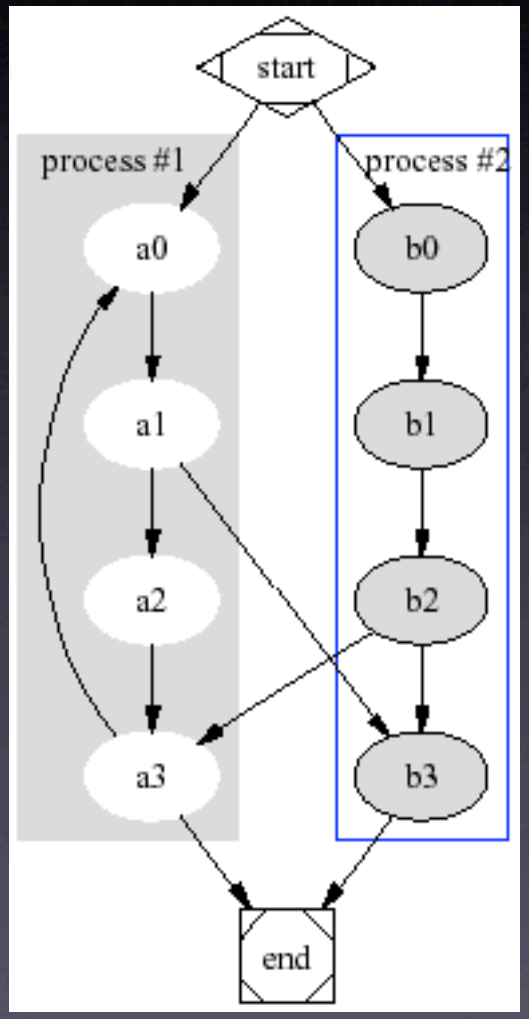


Records



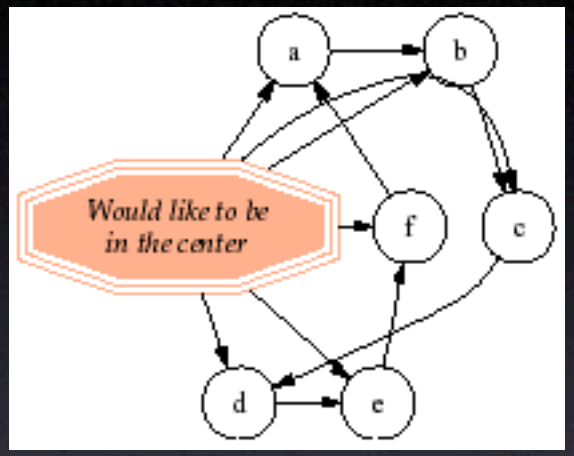
Binary Search Tree

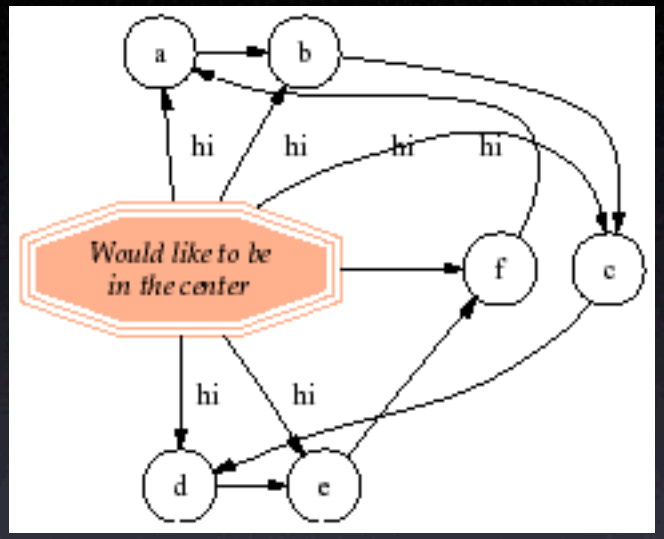
dot: Examples (cont.)

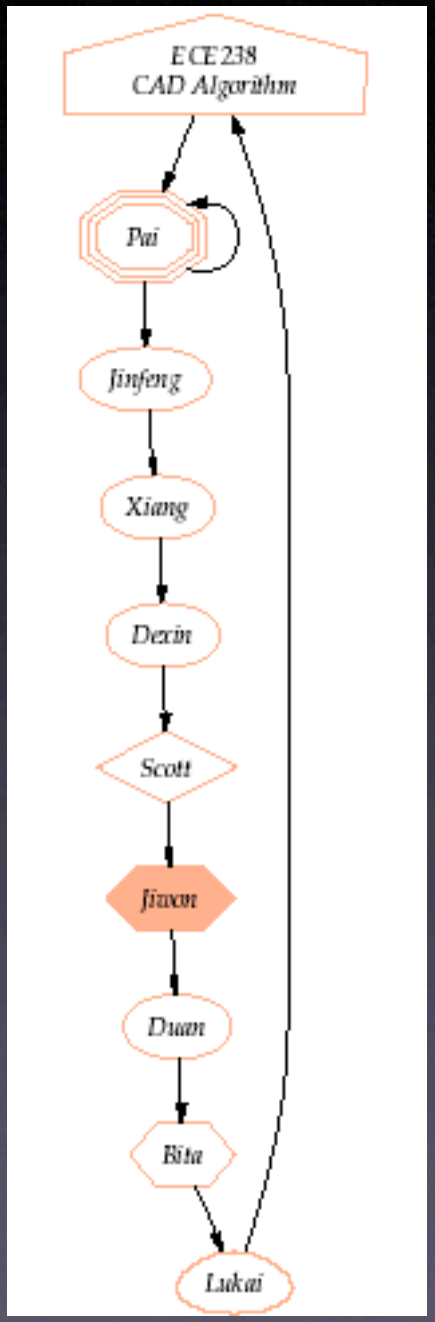


Process Diagram
with clusters

dot: My examples



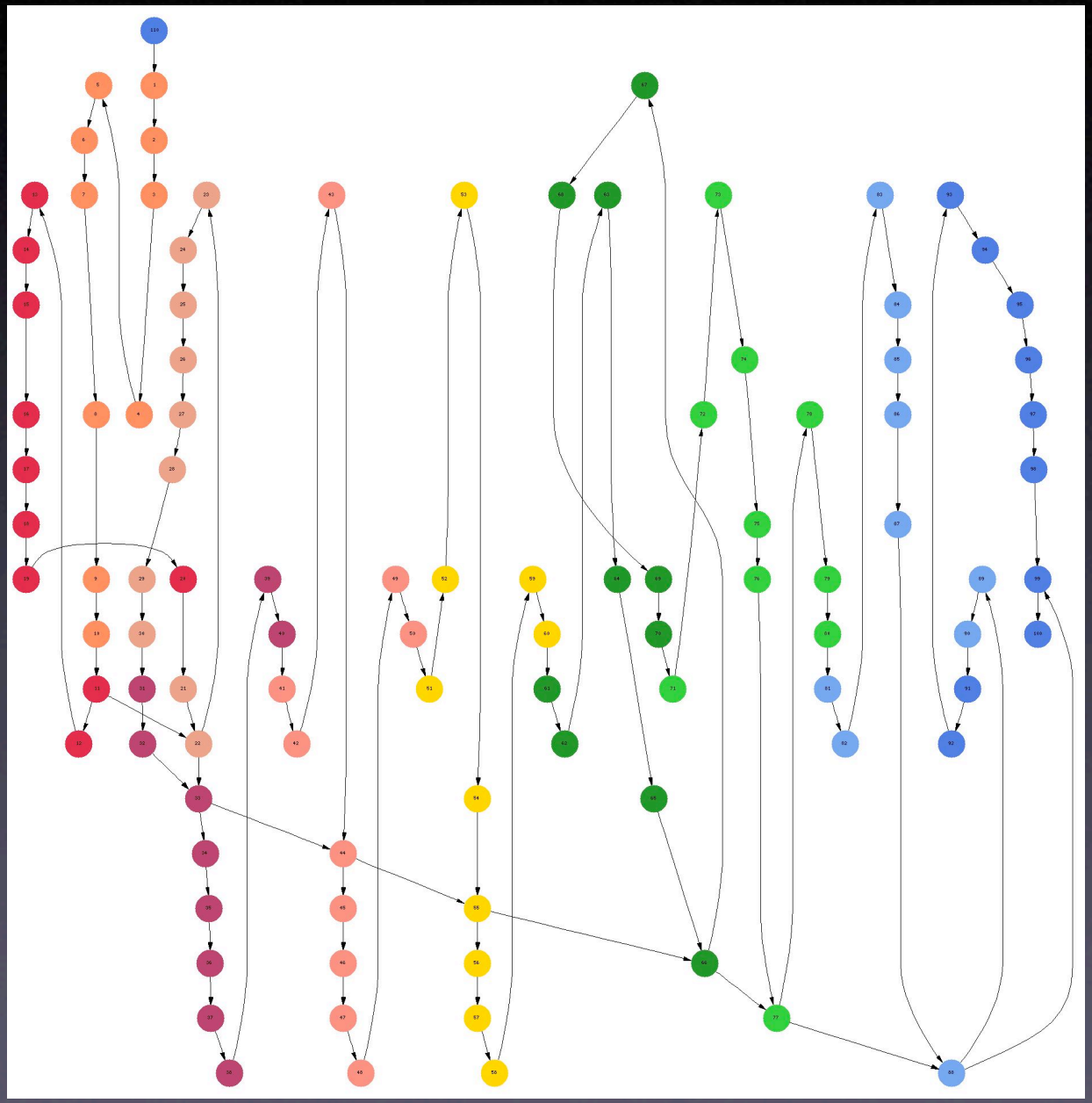




May 2003

Sun	Mon	Tue	Wed	Thu	Fri	Sat
-	-	-	-	1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

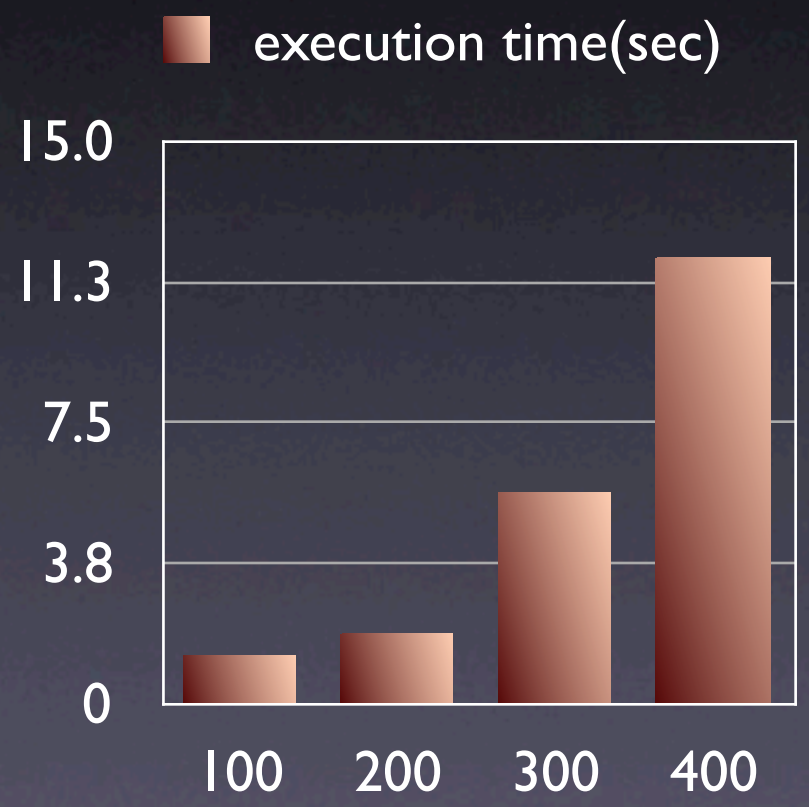
dot: My examples(cont.)



100 nodes
1.30 sec. user time
on a 700MHz
PowerPC

Trying Large Number of Nodes

- User time measured on PowerPC700
- Larger graphs representable
- Graph starts to look blurry at 200 nodes



Outline

- Introduction
- History
- DOT algorithm
  1. Ranking
  2. Ordering
  3. Positioning
  4. Drawing edges & labels
- Implementation (Graphviz)
- Conclusion

Summary

- dot's major contributions are:
  - application of network simplex for ranking and positioning
  - improved heuristic (median interpolation & local transposition) for reducing edge crossing
  - a method for making edge splines
- dot is...
  - straightforward to program (applied in many applications by now)
  - run fast, giving neat layout

References

- *DAG A Program that Draws Directed Graphs (1989): 50*
- *Application of Graph Visualization (1994): 26*
- *Drawing graphs with dot (2002): 39*
- *Drawing graphs with Graphviz(2003)*
- *R. Tamissa. Graph Drawing(1997)*
- *S. Haines, R. Kennaway, R. Sleep. Visualisation techniques for Event Stream Analysis*
- *L. Wong. A Protein Interaction Extraction System(2000)*

Drawing Graphs

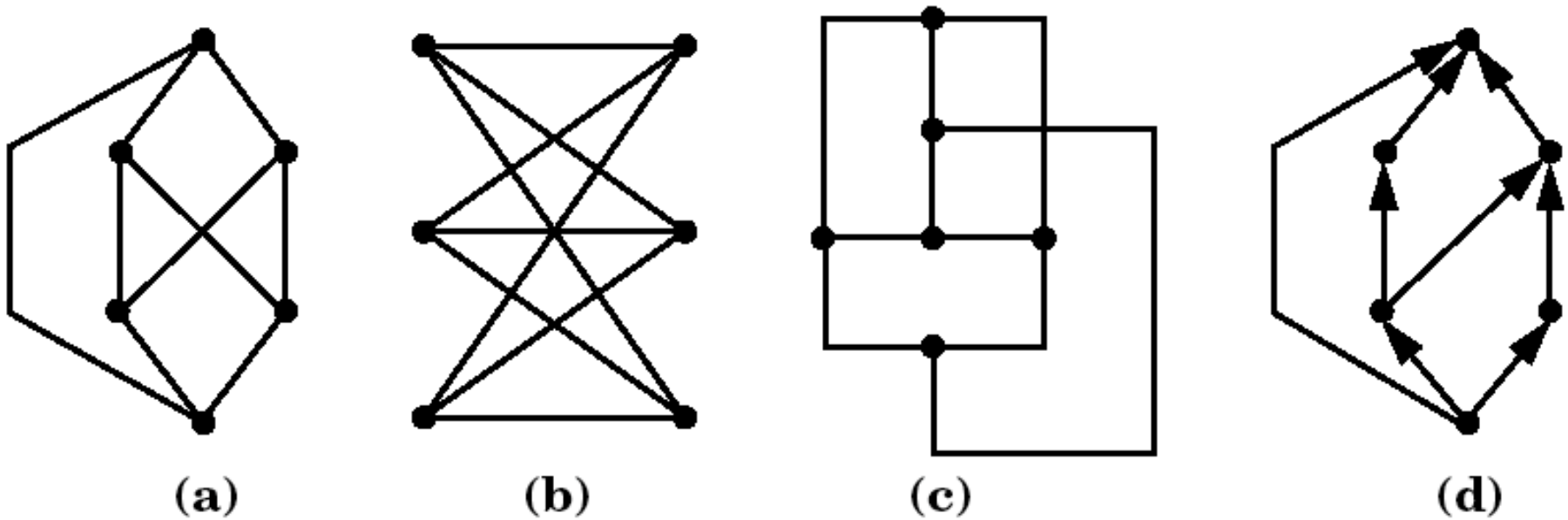
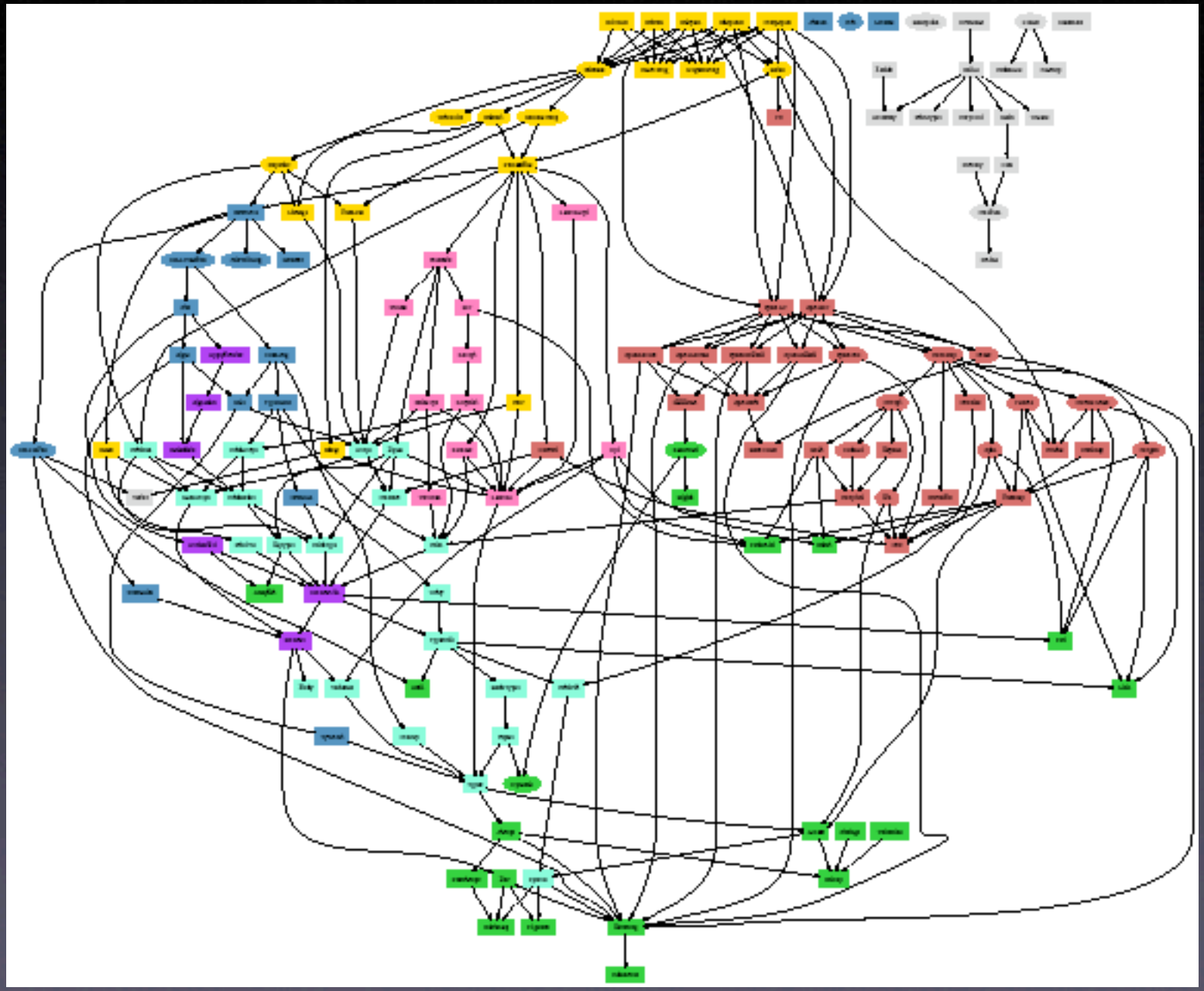
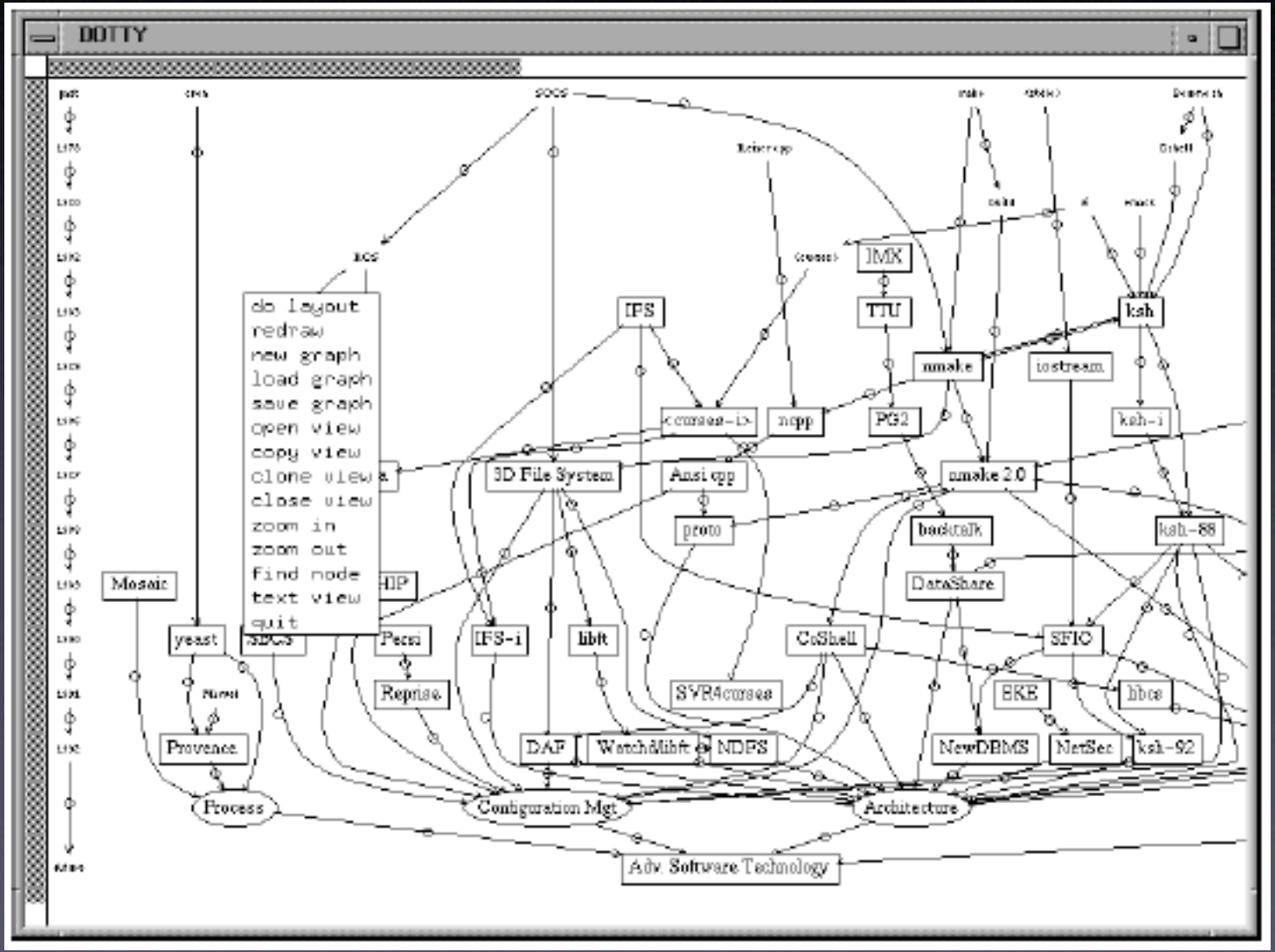


Figure 1.1: Types of drawings: (a) polyline drawing of $K_{3,3}$; (b) straight-line drawing of $K_{3,3}$; (c) orthogonal drawing of $K_{3,3}$; (d) planar upward drawing of an acyclic digraph.

dot: Example (cont.)



Dotty


DOTTY
do layout
redraw
new graph
load graph
save graph
open view
copy view
clone view
close view
zoom in
zoom out
find node
text view
quit
IMX
TTU
IFS
ksh
nmake
iostream
ncpp
PG2
ksh-i
3D File System
Ansi cpp
nmake 2.0
proto
backtalk
ksh-88
Mosaic
DataShare
yeast
IFS-i
libft
CoShell
SFIO
Reprise
SVR4curses
EKE
libcs
Provence
DAF
WatchMibft
NDPS
NewDBMS
NetSec
ksh-92
Process
Configuration Mgt
Architecture
Adv. Software Technology

Node ports

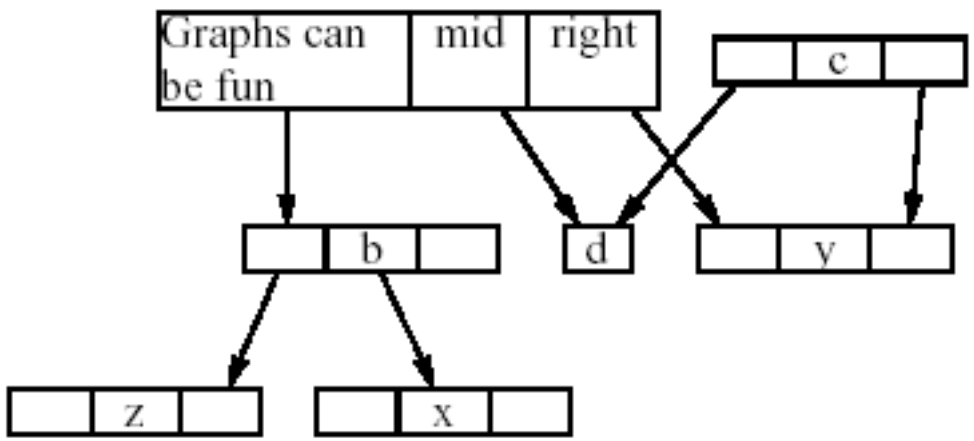


Figure 4-3. Node ports in a graph drawing