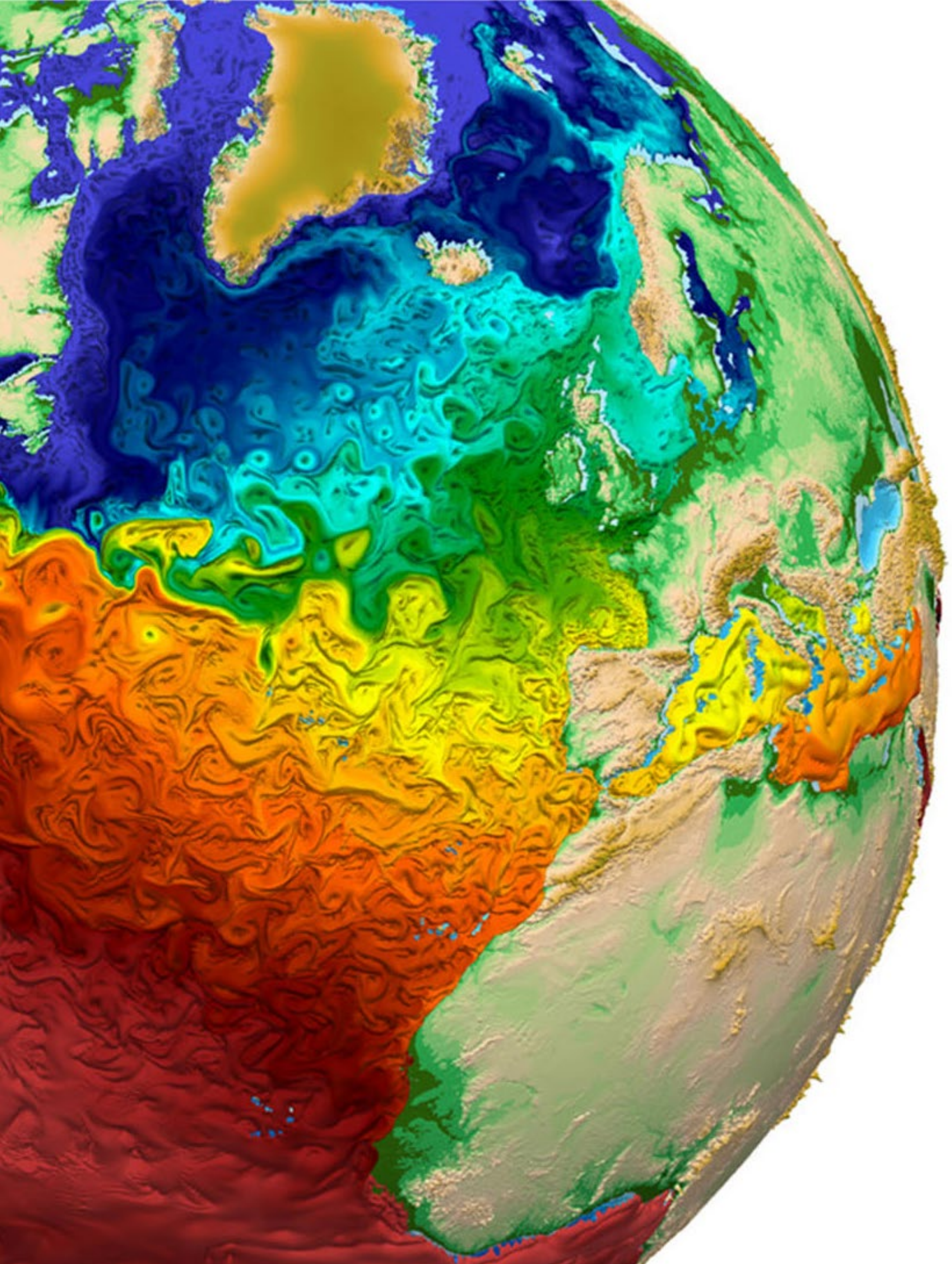


Forces-Directed Graph Layout

Scientific Visualization

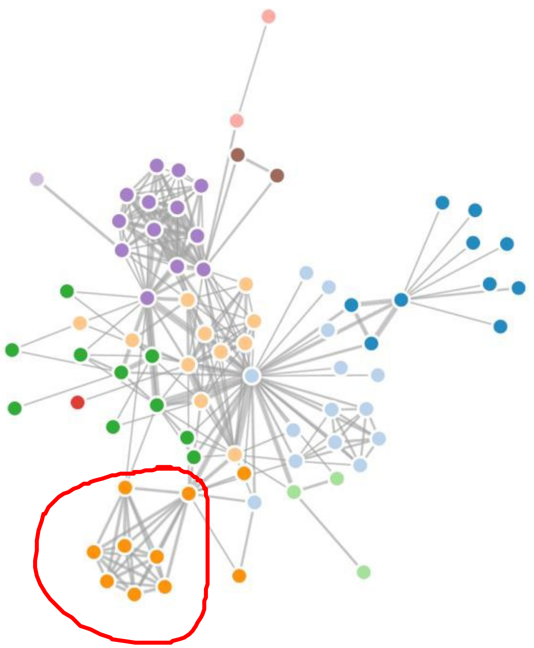
Professor Eric Shaffer

ILLINOIS

Force-Directed Graph Layout

Idiom: **force-directed placement**

- visual encoding
  - –link connection marks, node point marks
- considerations
  - –spatial position: no meaning directly encoded
    - left free to minimize crossings
  - –proximity semantics?
    - sometimes meaningful
    - sometimes arbitrary, artifact of layout algorithm
    - tension with length
      - –long edges more visually salient than short
- tasks
  - –explore topology; locate paths, clusters
- scalability
  - –node/edge density E < 4N



http://mbostock.github.com/d3/ex/force.html

Force-directed Layout Intuition

Goals

- Vertices well-distributed on the display
- Edges cross each other as little as possible.

Approach

- Simulate the graph as a physical system.
- Nodes are electrically charged particles and repulse each other
- Edges act as springs that attract connected nodes

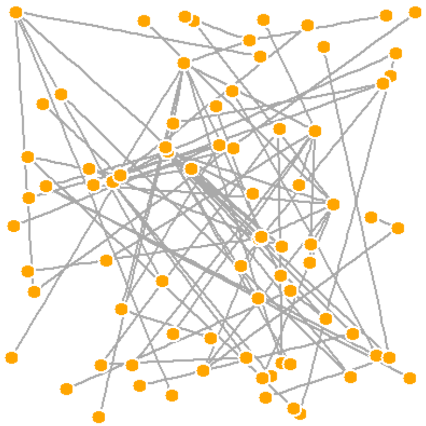
Result

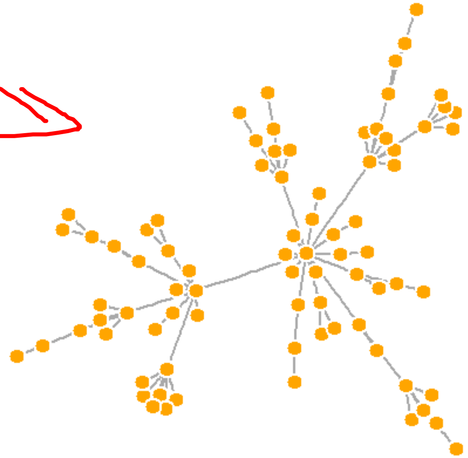
- Nodes are evenly distributed through the chart area
- Nodes which share more connections are closer to each other

"Fruchterman-Reingold is one of the most used force-directed layout algorithms out there." – The Internet

Graph Drawing by Force-Directed Placement Fruchterman, Thomas M. J.; Reingold, Edward M. (1991)

Developed at University of Illinois





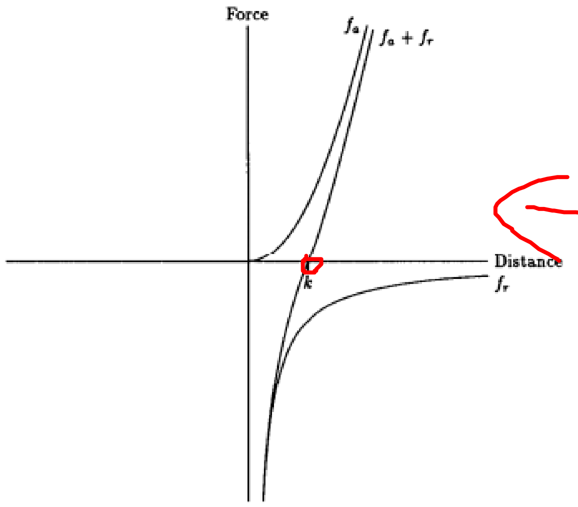
ILLINOIS

Fruchterman-Reingold Details

$$F_a(n_i, n_j) = \frac{|p_i - p_j|^2}{k}$$

$$F_r(n_i, n_j) = -\frac{k^2}{|p_i - p_j|}$$

$$k = C\sqrt{\frac{area}{number\ of\ vertices}}$$



n_i is a vertex

p_i is the position of that vertex

C is a constant found experimentally to yield good result

k if vertices are uniformly distributed,
would be radius of empty circle around vertex

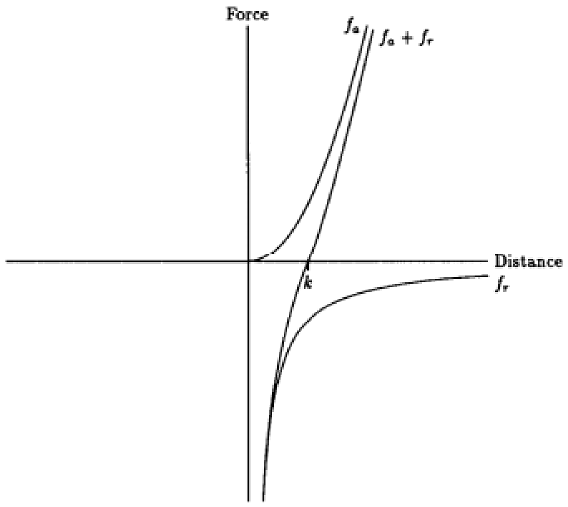
the distance at which the forces will balance

Fruchterman-Reingold Details

$$F_a(n_i, n_j) = \frac{|p_i - p_j|^2}{k}$$

$$F_r(n_i, n_j) = -\frac{k^2}{|p_i - p_j|}$$

$$k = C\sqrt{\frac{area}{number\ of\ vertices}}$$



For each vertex n_i calculate a sum of forces:

- $F_a(n_i, n_j)$ between n_i and all neighbors n_j
- $F_r(n_i, n_j)$ between n_i and all vertices n_j
- Each force has a magnitude and direction $\overleftarrow{p_j - p_i}$
- For repulsion, direction is negated
- Move p_i accordingly

Do this until change in positions below a threshold
…or you hit your limit on the number of iterations

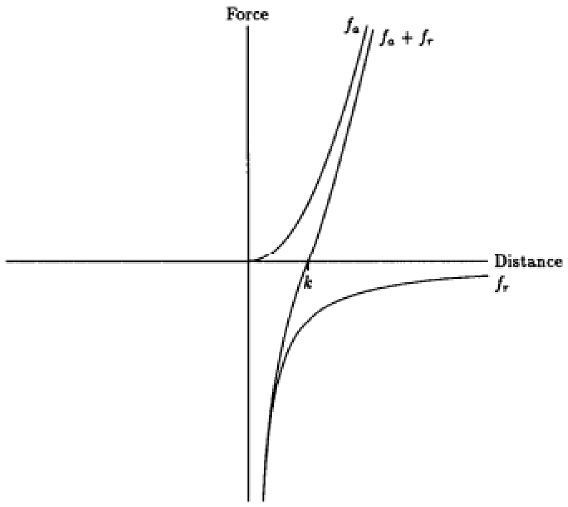
ILLINOIS

Fruchterman-Reingold…More Details

$F_a(n_i, n_j) = \frac{|p_i - p_j|^2}{k}$

$F_r(n_i, n_j) = -\frac{k^2}{|p_i - p_j|}$

$$k = C\sqrt{\frac{area}{number\ of\ vertices}}$$

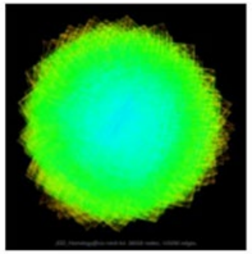


- Direction $\overleftarrow{p_j - p_i}$ should be unit length vector
- Each iteration, stop vertices from moving outside display
- Apply some scale factor to the amount of movement of each vertex
  - Reduce this scale factor each iteration

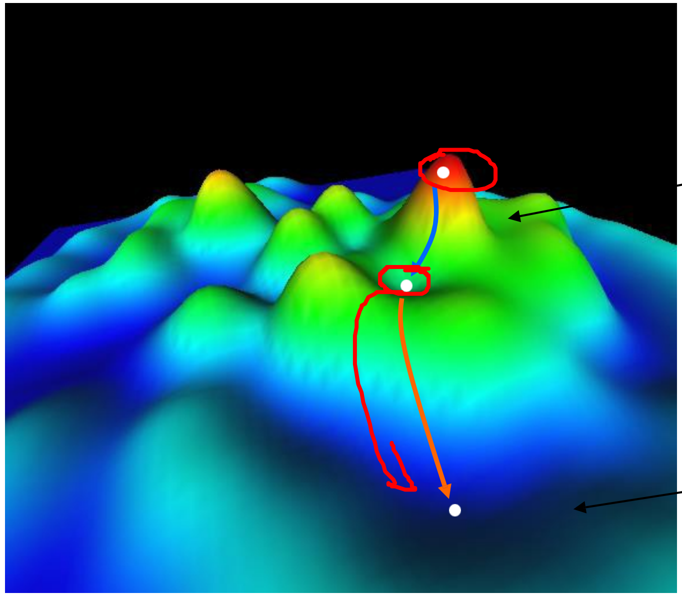
ILLINOIS

Disadvantages

- Computationally slow…$O(n^3)$
  - Can be sped up using spatial partitioning and calculating repulsion for only near-by nodes
  - Also can employ multi-level computation like Barnes-Hut and handle 1M node graph in seconds
- Visual limit is probably around 10K vertices/edges…hairball problem
- Layout can get trapped in locally optimal rather than globally optimal state



http://www.research.att.com/yifanhu/GALLERY/GRAPHS/index1.html



…minimizer may get stuck here (local minimum)

…instead of arriving here (global minimum)

Force-Directed Layout Advantages

- Simple to implement
- Can work on any graph (e.g. not just trees)
- Can change force functions to use other data for specific applications
  - e.g. directed edges, edge weights, different classes of nodes, etc.
- Can be interactive
- Works pretty well