Network Visualization

Hu “Efficient, High-Quality Force-Directed Graph Drawing”

Question: How to find a layout for network if nothing is known about its structural properties?

Requirements: flexibility, robustness, clarity

Approach: analogy to physics, i.e., nodes are objects, edges are interactions and forces

Goal: interconnected system at stable configuration = intuitively good layout

One of the solutions: force-directed methods

A force-directed method
1. models the graph drawing problem through a physical system of bodies with forces acting between them.
2. The algorithm finds a good placement of the bodies by minimizing the energy of the system.

Examples of forces to model
• Fruchterman, Reingold: system of springs between neighbors + repulsive electric forces
• Kamada, Kawai: springs between all vertices with spring length proportional to graph distance
**Force-directed methods**

Frequent problems that need to be addressed

1. **Many local minimums.** If we start with random configuration we can settle in one of the local minimums already after several iterations.

![Diagram showing a transformation from random configuration to a sphere, representing Fruchterman-Reingold algorithm.]

2. **Computational complexity.** Ideally we should model forces for all pairs of nodes. This gives us complexity $O(n^2)$ per iteration.


How to overcome these problems? Basic ideas: use multiscale algorithms and limit long-range forces.
**Force-directed methods**

$x_i \in \mathbb{R}^2$ or $\mathbb{R}^3$ - coordinates of node $i$

$\|x_i - x_j\|$ - 2-norm distance between $i$ and $j$

We define *spring-electrical* modes with two forces

- the repulsive force between any two nodes $i$ and $j$

  \[ f_r = -CK^2 / \|x_i - x_j\|, \ i \neq j \]

- the attractive force between any two neighbors $i$ and $j$

  \[ f_a = \|x_i - x_j\|^2 / K \]

The combined force on vertex $i$ is

\[
 f(i, x, K, C) = \sum_{i \neq j} -\frac{CK^2}{\|x_i - x_j\|^2}(x_j - x_i) + \sum_{ij \in E} \frac{\|x_i - x_j\|}{K}(x_j - x_i)
\]

Parameters (mostly for scaling): $K$ is spring length, $C$ strength of $f_a$ and $f_r$. Example: two connected nodes, $f$ is minimized when $\|x_i - x_j\| = KC^{1/3}$. 

*Introduction to Network Analysis*
Figure 2. Forces versus distance
Force-directed methods

The total energy of the system is

\[
\text{Energy}_{se}(x, K, C) = \sum_{i \in V} f^2(i, x, K, C)
\]

**Theorem 1.** Let \( x^* = \{x_i^* \mid i \in V \} \) minimizes the energy of the spring-electrical model \( \text{Energy}_{se}(x, K, C) \), then \( sx^* \) minimizes \( \text{Energy}_{se}(x, K', C') \), where \( s = (K'/K)(C'/C)^{1/3} \). Here \( K, C, K' \) and \( C' \) are all positive real numbers.
**Proof:** This follows simply by the relationship

\[
f(i, x, K, C) = \sum_{i \neq j} \frac{-C K^2}{\|x_i - x_j\|^2} (x_j - x_i) + \sum_{i \leftrightarrow j} \frac{\|x_i - x_j\|}{K} (x_j - x_i)
\]

\[
= \left(\frac{C}{C'}\right)^{2/3} \frac{K}{K'} \left( \sum_{i \neq j} \frac{-C' (K')^2}{\|s x_i - s x_j\|^2} (s x_j - s x_i) + \sum_{i \leftrightarrow j} \frac{\|s x_i - s x_j\|}{K'} (s x_j - s x_i) \right)
\]

\[
= \left(\frac{C}{C'}\right)^{2/3} \frac{K}{K'} f(i, s x, K', C')
\]

where \(s = (K'/K)(C'/C)^{1/3}\). Thus,

\[
\text{Energy}_{se}(x, K, C) = \left(\frac{C}{C'}\right)^{4/3} \left(\frac{K}{K'}\right)^2 \text{Energy}_{se}(s x, K', C').
\]
**Force-directed methods**

Another example Kamada-Kawai *spring* model

- the repulsive force between any two nodes $i$ and $j$

\[
  f_r(i, j) = f_a(i, j) = \| x_i - x_j \| - d(i, j), \ i \neq j
\]

The combined energy of the system is

\[
  \text{Energy}_s(x) = \sum_{i \neq j} (\| x_i - x_j \| - d(i, j))^2
\]
Introduction to Network Analysis

- ForceDirectedAlgorithm(G, x, tol) {
  - converged = FALSE;
  - step = initial step length;
  - Energy = Infinity
  - while (converged equals FALSE) {
    * x^0 = x;
    * Energy^0 = Energy; Energy = 0;
    * for i ∈ V {
      . f = 0;
      . for (j ↔ i) f := f + \frac{f_k(i,j)}{\|x_j-x_i\|} (x_j - x_i);
      . for (j ≠ i, j ∈ V) f := f + \frac{f_r(i,j)}{\|x_j-x_i\|} (x_j - x_i);
      . x_i := x_i + step * (f / \| f \|);
      . Energy := Energy + \| f \|^2;
    *
    }
    * step := update_steplength(step, Energy, Energy^0);
    * if (\| x - x^0 \| < K tol) converged = TRUE;
  - }
  - return x;
- }

Algorithm 1. An iterative force-directed algorithm.
function update_steplength (step, Energy, Energy⁰)
    if (Energy < Energy⁰) {
        progress = progress + 1;
        if (progress >= 5) {
            progress = 0;
            step := step / t;
        }
    } else {
        progress = 0;
        step := t step;
    }

step := t step

Best minimized layout

70 iterations
The repulsive force calculation resembles the $n$-body problem in physics, which is well studied. One of the widely used techniques to calculate the repulsive forces in $O(n \log n)$ time with good accuracy, but without ignoring long-range forces, is to treat groups of faraway vertices as supernodes, using a suitable data Structure.
function MultilevelLayout \( (G^i, \text{tol}) \)

- Coarsest graph layout
  - if \((n^{i+1} < \text{MinSize or } n^{i+1} / n^i > \rho)\) {
    * \(x^i\) = random initial layout
    * \(x^i = \text{ForceDirectedAlgorithm}(G^i, x^i, \text{tol})\)
    * return \(x^i\)
  - }

- The coarsening phase:
  - set up the \(n^i \times n^{i+1}\) prolongation matrix \(P^i\)
  - \(G^{i+1} = P^i \cdot G^i \cdot P^i\)
  - \(x^{i+1} = \text{MultilevelLayout}(G^{i+1}, \text{tol})\)

- The prolongation and refinement phase:
  - prolongate to get initial layout: \(x^i = P^i \cdot x^{i+1}\)
  - refinement: \(x^i = \text{ForceDirectAlgorithm}(G^i, x^i, \text{tol})\)
  - return \(x^i\)

**Algorithm 2.** A multilevel force-directed algorithm.
http://www.cise.ufl.edu/research/sparse/matrices/
High-dimensional Embedding
(see paper [KH])

Algorithm

- Choose \( m \) pivots \( \{p_1, \ldots, p_m\} \)
- Each \( v \in V \) is associated with \( m \) coordinates
  \[
  \{X^i(v)\}_{i=1}^m, \quad \text{where} \quad X^i(v) = d(p_i, v)
  \]
- Project \( m \)-dimensional coordinates into 2- or 3-dimensional space

How to choose \( p_i \)

- choose \( p_1 \) at random
- For \( j = 2, \ldots, m \) choose \( p_j \) that maximizes the shortest distance from \( \{p_k\}_{k=1}^{j-1} \)

Similar to the \( k \)-center problem where the goal is to minimize the distance from \( V \) to \( k \) centers.