Bioinformatics III

Prof. Dr. Volkhard Helms — Saarland University
Ruslan Akulenko, Ahmad Barghash, Duy Nguyen, Christian Spaniol — Chair for Computational Biology
Winter Semester 2013/2014

Exercise Sheet 2

Due: November 8, 2013 13:15

Submit your solutions on paper, hand-written or printed at the *beginning* of the lecture or in building E2 1, Room 3.01. Alternatively you may send an email with a single PDF attachment. If possible, please include source code listings. Additionally hand in all source code via mail to christian.spaniol@bioinformatik.uni-saarland.de.

2 More On Networks, Forces, and Force Directed Layouts

We continue to evolve our network software from the first assignment. As announced, the assignment of this week deals with the construction of scale-free networks.

Then we will discuss energies and forces, and finally use force calculations to layout some real networks.

This is a two weeks assignment. For those who find themselves done within a week, there is a bonus exercise on suffix primes to collect additional points.

Exercise 2.1: The Scale-Free Network (30pts)

First, (a) construct a scale-free network according to the Barabási-Albert model. Then (b) examine the degree distribution of such networks and determine some characteristics. Finally, (c) compare the characteristics of random networks and scale-free networks.

(a) Implement the algorithm given in the lecture to set up a scale-free network according to the Barabási-Albert model. Start from the first three connected nodes and add each new node with two links. Connect the new links with increasing preference to nodes that have higher degrees. The class can be derived from the abstract network class that you wrote in the first assignment.

- To extend our previous network class, the semantics of our abstract network initializer have to change. Now, the parameters specify the *number of nodes to be added* and the *number of links that are added each iteration step*.
- Think of a method to map probabilities rather than to compute them directly.

(b) Determine the degree distribution for a scale-free network. Plot the degree distribution for a network of 10 000 and 100 000 nodes, respectively, with double logarithmic axes.

(c) Create a random network with the same number of nodes and links as the scale-free network of 100 000 nodes and plot the degree distributions of both networks into the same plot.

For which combination of logarithmic and/or linear axes is the difference between the degree distributions of the two networks seen best?

Identify and explain the two major differences between the degree distribution of the random and of the scale-free network.

Exercise 2.2: Energy and forces (20pts)

(a) **Configuration of minimal energy:**

Determine the equilibrium distance between two equally charged mass points which are connected by a spring. At the equilibrium distance the total force vanishes. Verify that

instead of calculating the forces explicitly, it is equivalent to determine the configuration of minimal energy.

Hints:

- The force equals the negative gradient of the energy, i.e., the force is a measure for how much the energy changes with an infinitesimal displacement:

$$\overrightarrow{F}(\overrightarrow{r}) = -\nabla E(\overrightarrow{r}), \text{ with the gradient operator } \nabla := \begin{pmatrix} \mathrm{d}/\mathrm{d}x \\ \mathrm{d}/\mathrm{d}y \\ \mathrm{d}/\mathrm{d}z \end{pmatrix}$$

  In a single dimension, this reduces to $\nabla = \mathrm{d}/\mathrm{d}r$, i.e. the simple derivative with respect to the distance r. The gradient of a function can consequently be understood as a multidimensional slope.

- The interaction energy between two charges q_1 and q_2 is given as:

$$E_c(r) = \frac{1}{4\pi \cdot \epsilon_0 \epsilon} \frac{q_1 q_2}{r}$$

  For the connecting spring use the harmonic potential:

$$E_h(r) = \frac{k r^2}{2}$$

- To show the equivalence of vanishing force and minimal energy remember how the minimum of a function is defined. Also note that the distance between two particles is a one-dimensional measure.

(b) **Force field from a spherically symmetric potential:**

Calculate the force fields $\overrightarrow{F}(\overrightarrow{r}) = -\nabla E(\overrightarrow{r})$ for both the Coulomb interaction E_c and the harmonic potential E_h in cartesian coordinates.

Hints:

- Write ∇ and the resulting force field $\overrightarrow{F}(\overrightarrow{r})$ in component form. Then you get one equation for x, y, and z, each. This is the form that you need for the second part.
- Note that:

$$r = \sqrt{x^2 + y^2 + z^2}, \qquad \overrightarrow{F}(\overrightarrow{r}) = \begin{pmatrix} F_x(x) \\ F_y(y) \\ F_z(z) \end{pmatrix}$$

Exercise 2.3: Force directed layout of graphs (50pts)

Implement an algorithm to layout networks, using energy functions to mimic the repulsive and attractive behaviour. Subsequently, read networks from files and visualize energy trajectories and the final layout.

(a) Between all nodes, use a repulsive degree dependent Coulomb type potential, defined as:

$$E_c(r_{ij}) = \frac{k_i \cdot k_j}{r_{ij}}$$

Additionally, for interacting nodes, use a degree independent harmonic attractive potential:

$$E_h(r_{ij}) = \frac{r_{ij}^2}{2}$$

The parameter r_{ij} is the distance between two nodes i and j. Because we layout in 2D, the squared distance is defined as:

$$r_{ij}^2 = (x_i - x_j)^2 + (y_i - y_j)^2$$

The interaction between two nodes i, j is defined as:

$$W[i][j] = \begin{cases} 1, & \text{if edge } i \to j \text{ exists} \\ 0, & \text{else} \end{cases}$$

Implement a layouter class to layout graphs by force interactions.

Hints:

- In order store energy and positions in the respective node objects, you can extend your class *"on-the-fly"*, i.e. node.force_x, node.force_y, node.pos_x, node.pox_y = 0.0, 0.0, 0.0, 0.0.
- For each node, choose an initial position on a two-dimensional plane. A reasonable start is within $\pm 10 - 20$ units from the origin.
- To layout the graph, perform at least $500 - 5000$ iteration steps of the following algorithm:
  (1) Calculate the pairwise forces between all nodes and sum them up for each of the nodes:
  $$\overrightarrow{F}_{ij} = \overrightarrow{F}_c(\overrightarrow{r}_{ij}) + W[i][j] \cdot \overrightarrow{F}_h(\overrightarrow{r}_{ij})$$
  Thus, the total force on node i is $F_i = \sum_j F_{ij}$. Note that the forces between two nodes are symmetric, i.e., $F_{ij} = -F_{ij}$.
  (2) Add a random force in the range $[-0.3; 0.3]$ to the total force on each node. This additional "thermal" contribution improves the convergence behavior and helps to escape local minima.
  (3) Update the position of each node from the forces as:
  $$\Delta r_i = \alpha \cdot F_i$$
  A reasonable value is $\alpha = 0.03$. Do not forget to reset all the forces after this step.
  (4) Calculate the total energy, which is the sum of all individual interaction energies:
  $$E_{\text{tot}} = \sum_{j>i} E_c(r_{ij}) + W[i][j] \cdot E_h(r_{ij})$$
  Print out this energy together with the iteration number to a file.
- To output your layout: if you are using *Gnuplot*, then loop over all edges and print the x and y positions of the two nodes of a link on a *separate* line, each, followed by an empty line. Plot the resulting file "`with linespoints`" to get a representation of the layed out graph.

(b) Proceed to actual network layouts. Therefore, implement a network class to import networks from files:

```
class GenericNetwork(AbstractNetwork):
  def __init__(self, filename):
  """
  Create a network from a file that contains a list of node ids, e.g.
  0 1
  0 2
  1 2
  ...
  """
```

(c) Start with the test files "`star.txt`", "`square.txt`", "`star++.txt`" and "`dog.txt`", which are available on our homepage. The resulting final configurations should converge at increasing energy levels. Simple networks need about 300 iterations, larger networks may take up to a thousand iterations. Run the layout algorithm multiple times if necessary.

Exercise 2.4: Suffix primes (Bonus exercise, 30pts)

A suffix prime is a prime of which each suffix again is a prime. The following number is the longest suffix prime, for example:

$$357686312646216567629137$$

A complete list of suffix primes is given in the supplementary material.

(a) Extend your network to an `SuffixPrimeNetwork`-class in order to resemble a *undirected* network of suffix primes. A node with identifier (the prime) s and length n is connected to all nodes that have a identifier length of $n+1$ and $n-1$ with the respective identifiers that match $s' :: s$ and $:: s$. In other words, extend the network, so that level one nodes contain two digits, level two nodes contain three digits, and so on. For a better understanding, see figure 1 which resembles a similar datastructure.

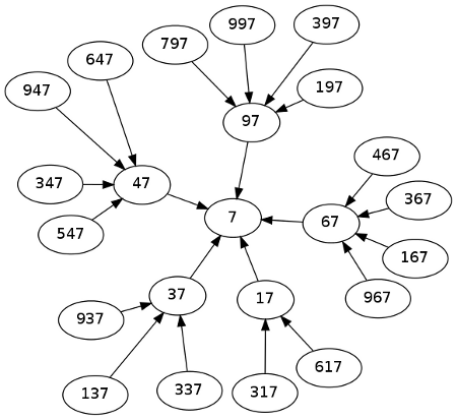

Figure 1: An excerpt of a suffix prime tree, nodes only shown up to level two.

(b) How many individual components does your network consist of? What is their size?

(c) Determine and plot the degree distribution. Explain your findings.

(d) Determine and plot the level distribution, i.e. how many nodes share the same level. Try to explain.

Hint:

- The length of the longest suffix prime resembles the highest level. You could init a list of that length containing "0" 's in order to keep track of the node amount of nodes within a certain level and increase them as you create your network.

(e) Use the layouter you wrote in the previous exercise to layout the suffix prime network up to level 4, including. How would you characterize the network? The network contains more than a single cluster. What happens to the individual clusters? Explain why.

Have fun!