Graph Analysis

For this implementation project, you may use networkx in Python or igraph in Python and R.

Network Properties for Ecoli

Download the file Ecoli.txt that gives the gene network for E.Coli. It has an (undirected) edge whenever there is an interaction between the two genes.

Write an script to compute several of the graph properties for the E.coli network:

- *Diameter*: Compute the diameter. Note that normally, d_{ij}, the distance (shortest path length) between node i and j, is considered to be infinity if i cannot reach j. For this assignment, simply ignore d_{ij} if i cannot reach j. In other words, compute the diameter as the maximum d_{ij} over all reachable pairs i and j. *Note*: do not use the builtin *diameter* function in networkx/igraph. However you may compute the shortest paths using the *shortest paths* functions.
- *Degree Distribution*: Print the degree distribution of the Ecoli network. Next, plot the degree k and the probability $P(k)$ in a log-log plot to see whether the plot looks like a straight line. Use the linear regression function to fit a line to the log-log data, and plot the line and print its slope, corresponding to the exponent in the power-law degree distribution: $P(k) \propto k^{-\gamma}$. You may omit the first few and last few degree values to get a better fit. Use your judgement to get a "good" fit. *Note*: do not use the builtin degree distribution related functions.
- *Clustering Coefficient*: Compute the clustering coefficient for the graph. If a node has degree less than 2, assume that its local clustering coefficient is 0. *Note*: do not use the builtin *transitivity*, or *clustering coefficient*, or related functions.

Page Rank in E. Coli

Download the file Ecoli-directed.txt that gives the gene regulatory network for E.Coli. It has a directed edge from gene u to gene v, if u controls v. Create a directed graph from these edges.

Compute the pagerank for each gene, via the power-iteration method, assuming that $d = 0.1$, that is with probability 0.1 we follow a random link, and with probability 0.9 an existing link from each node. That is write a function to find the dominant eigenvector and eigenvalue for the pagerank matrix of the directed Ecoli graph. One can compute the dominant eigen-vector/-value of a matrix $\mathbf{M}$ iteratively as follows.

Let $\mathbf{x}_0 = \begin{pmatrix} 1 \\ 1 \\ \\ \vdots \\ 1 \end{pmatrix}$ be the starting vector. In each iteration i, we compute the new vector:

$$\mathbf{x}_i = \mathbf{M}\ \mathbf{x}_{i-1}$$

We then find the element of $\mathbf{x}_i$ that has the maximum absolute value, say at index m. For the next round, to avoid numerical issues with large values, we re-scale $\mathbf{x}_i$ by dividing all elements by $\mathbf{x}_{im}$, so that the largest value is always 1 before we begin the next iteration. To test convergence, you may test the magnitude of the difference between the old and new scaled vector, i.e., $\|\mathbf{x}_i - \mathbf{x}_{i-1}\|$, and you can stop if the difference between these two falls below some threshold, say 0.001.

Run your iterative dominant eigensolver on the Ecoli-directed dataset. Find the dominant eigen-value and eigen-vector. For the final eigen-vector, make sure to normalize it, so that it has unit length. Print the top 10 genes according to pagerank. Note: do not use the builtin **page rank** or **eigen** methods.