

Erdos-Reyni Random Graphs in MATLAB

Farnaz Ronaghi

April 20, 2010

1 Prerequisites and Bug Fixes

In this tutorial, we'll look at generating Erdos-Reyni random graphs in Matlab. First we start with a list of prerequisites and step-by-step guide on how to fulfill them. Random graph visualizations rely on the AT&T GraphViz program, MATLAB interface to AT&T GraphViz program and MESHPART toolbox. You should first download these packages to your computer.

You can start by installing GraphViz on your machine. Based on the operating system in use, installation steps might be different. You Please check out application's README file for detailed installation notes. Please note that your MATLAB should either be closed during GraphViz installation or you should restart it afterwards. You can now uncompress MATLAB interface to GraphViz and MESHPART to a target directory. After uncompressing, you have to add the target directory to MATLAB path. Please note that the target directory should contain a couple of MATLAB files. To add a directory to MATLAB path you can use *path* command in MATLAB. If you don't know how to use this function, you can type *help path* in your MATLAB command window.

You have everything installed now but not yet ready to start due to authentic errors in GraphViz interface and functional updates in MESHPART toolbox. To solve these problems, first go to the target directory for GraphViz interface. Open the file named, *dot_to_graph.m*. You should change the scanned data type to float at line **92** or you can simply replace the specified line with the following:

```
[node_pos]=sscanf(line(pos_pos:length(line)), ' pos ="%f,%f"');
```

There's one other minor bug in the file named *draw_dot.m*. You should change line **41** to the following command:

```
labels = cellstr(num2str(num_names(lbl_idx)'));
```

Done with the bugs in MATLAB interface we can now start our visualizations.

2 Giant Component

In this section we will generate a random graph, visualize it and look at the giant connected component. Given the number of vertices and probability of an edge as n and p , we can instantiate a random graph, G . G basically shows the adjacency matrix for the graph. The following sets of commands generates a random graph G , makes it upper-triangular and then makes it symmetric.

```
G = rand(n, n) < p;
```

```
G = triu(G, 1);
```

```
G = G + G';
```

From the theory of Erdos-Reyni graphs, we know that a giant connected component should emerge when $p > 1/(n - 1)$. By setting p to be greater than this value we can generate random graphs with a giant connected component. Using the Matlab interface to the AT&T Graphvis program *neato*, we can generate a layout for this graph and visualize it using the *matlab gplot* function. *draw_dot*, interacts with GraphViz *neato* program and generates a layout including coordinates of each graph node. Depending on input size, it might take a while for *neato* to generate layout files. *gplot* takes the graph adjacency matrix and node coordinates as input and plots them. Now, we can see if the graph really does have a giant component.

If you run *giant_component.m*, you can see the large connected component of the graph in drawn figure. GraphViz deletes single nodes with no links by default, so if you are wondering why the number of nodes on the screen is smaller than what you entered, this should be the case.

3 Evolution

Now that we know how to generate Erdos-Reyni random graphs, let's look at how they evolve in p – the probability of an edge between two nodes. If we keep adding edges to the graph randomly, we expect to see a giant component with approximately $\log(n)$ vertices emerge when p is near $\frac{1}{(n-1)}$. Then, we expect the graph to become completely connected when p is close to $\frac{\log(n)}{n}$.

We should first setup our graph as a function of p , we can use anonymous functions in Matlab 7 to do this. The following first three commands generate a random, symmetric matrix. The last command defines G as a function of p , it basically generates a link between two vertices with probability p .

```
A = rand(n, n);
A = triu(A, 1);
A = A + A';
G = @(p)A < p;
```

using the AT&T GraphViz tool and MATLAB gplot, we can plot each graph. You can set p to be equal to $\frac{1}{(n-1)}$, the graph might not have a giant connected component. Remember that we aren't guaranteed to find a giant component at this p , but it has to happen soon. In this case, notice that it is very likely that the two larger components will merge together very soon.

To animate the results for different values of p , first we generate a basic layout and draw all the graphs based on that. In practice the geometric mean of $\frac{1}{(n-1)}$ and $\frac{\log(n)}{n}$ gives a good layout for drawing graphs. The following command takes care of that:
`[xy] = draw_dot(G(sqrt(thres_conn * thres_giant)))`

evolution.m script runs the animation and plots the size of the largest connected component. Connected component calculations are handled by MESHPART toolbox. The function *components.m* from MESHPART toolbox, calculates component information. The animation pauses briefly at the component where a giant threshold emerges. You can run the animation with different graph sizes and find the phase transitions on plots. To change animation speed you can use the delay parameter, setting it to numbers greater than one increases the delay and vise versa.