SAND2021-12359PE

Sandia National Laboratories

Exceptional service in the national interest

Barabási-Albert Algorithm

Abigail Pribisova & Robert Garrett

GRAPH THEORY & APPLICATION COURSE 2021

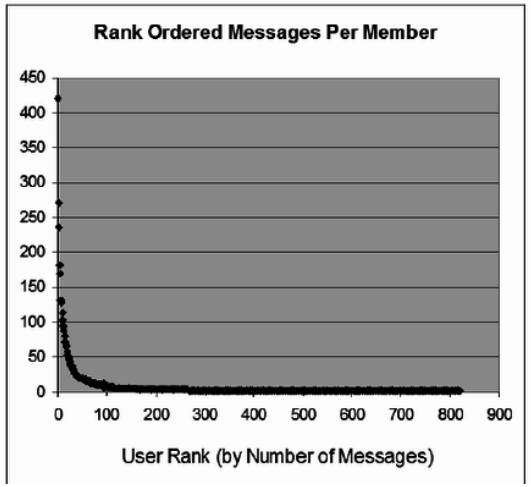
Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia LLC, a wholly owned subsidiary of Honeywell International Inc. for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

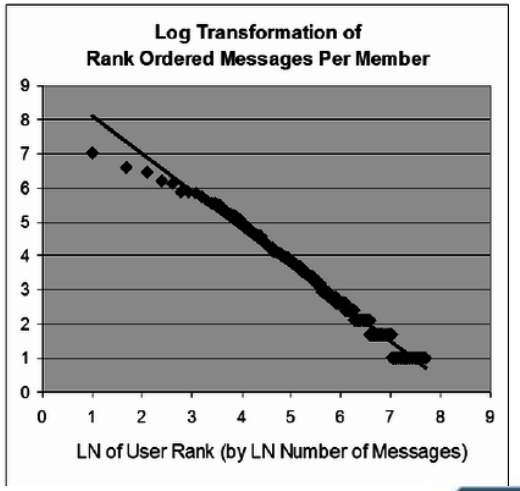
Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

Overview

The Barabási-Albert (BA) model is an algorithm for generating random scale-free networks using preferential attachment.

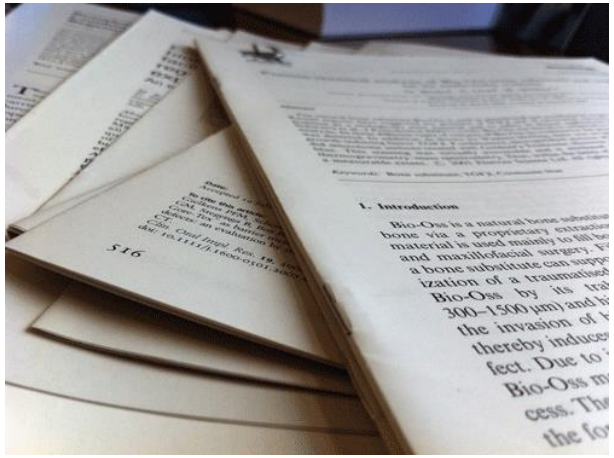
- network: set of nodes that are connected together with edges
- random: links (or edges between nodes) are chosen completely at random
- scale-free: degree distribution follows a power law
  - degree distribution: representation of the number of connections each node has to other nodes for all the nodes in the network
  - power law: functional relationship between two quantities where one quantity varies as a power of the other
  - log scale x and y will produce a **straight line**
  - i.e. scale-free: a small number of the nodes have extremely small or large degrees
- preferential attachment: nodes with high degrees are more likely to have more nodes connected to them





History

- Developed in 1999 by Albert-László Barabási and Réka Albert
- Special case of Price model developed by Derek de Solla Price in 1976
  - Network of citation between scientific papers – used a directed network with “cumulative advantage” (or preferential attachment)
  - Well-cited papers get cited more often!
- Undirected version -> BA model!
  - Degree distributions on the web
  - There are only a couple of VERY frequently visited sites (like Google), and most sites will connect to them



Algorithm

1. Start with a connected network of *n* nodes
2. Add nodes one at a time
   1. Each node is connected to *m* existing nodes with a probability proportional to the number of edges that the existing nodes already have
3. Stop when you have reached desired number of nodes

$$p_i = \frac{k_i}{\sum_j k_j},$$

- p_i: probability of new node being connected to node *i*
- k_i: degree of node *i*
- sum is over all pre-existing nodes *j*

Code for Algorithm

```
%%capture
!pip install networkx
!pip install matplotlib
```

```
import networkx as nx
import matplotlib.pyplot as plt
import random
```

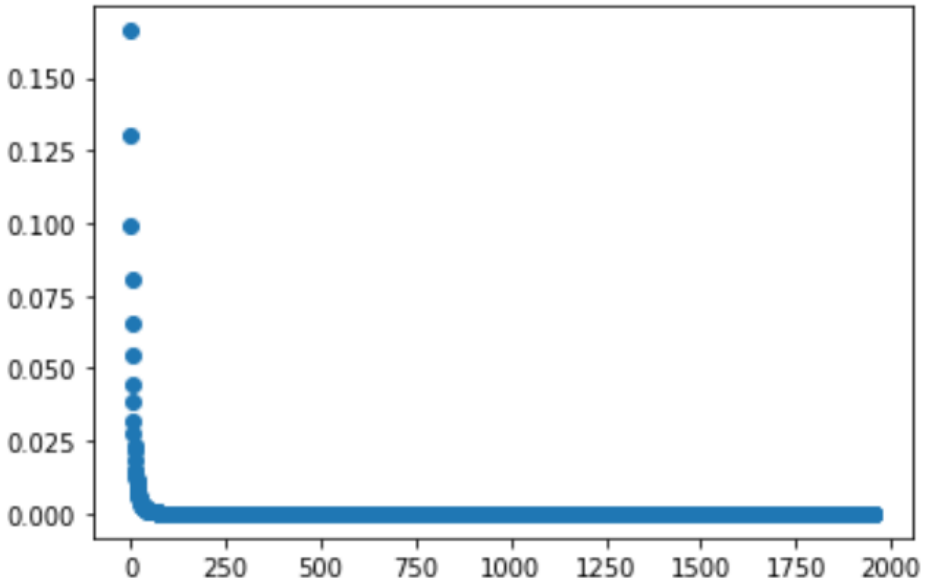
```
def barabasi_albert_net(init_num_nodes, num_edges_per_new_node, max_nodes):
    # this creates our initial connected graph, which is a complete graph
    G = nx.complete_graph(init_num_nodes)
    # initially, each node will have (init_num_nodes - 1) connections
    current_nodes = [val for val in range(init_num_nodes) for _ in range(init_num_nodes-1)]
    # add new nodes one at a time
    for i in range(init_num_nodes, max_nodes):
        # randomly choose a number of nodes to connect the new node to
        sampled_nodes = list(random.sample(current_nodes, num_edges_per_new_node))
        G.add_edges_from(zip([i] * num_edges_per_new_node, sampled_nodes))
        # current_nodes updated to include frequency of each node corresponding to degree
        current_nodes.extend(sampled_nodes)
        current_nodes.extend([i] * num_edges_per_new_node)
    # display graph
    nx.draw(G, with_labels=True)
    plt.show()
    # return graph for degree distribution plotting
    return G
```

Code for Plotting Degree Distribution and Result

```python
def plot_degree_distribution(graph):
    degree = nx.degree_histogram(graph)
    x = range(len(degree))
    y = sorted([z/float(sum(degree)) for z in degree], reverse=True)
    plt.scatter(x, y)
    plt.show()
```

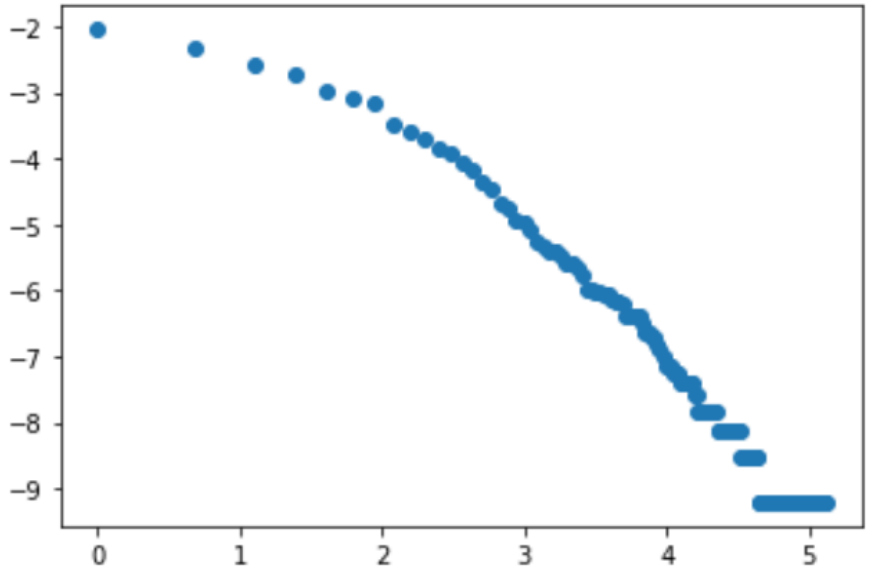
```python
G = barabasi_albert_net(10, 10, 100000)
```

```python
plot_degree_distribution(G)
```




```python
G = barabasi_albert_net(10, 10, 10000)
```

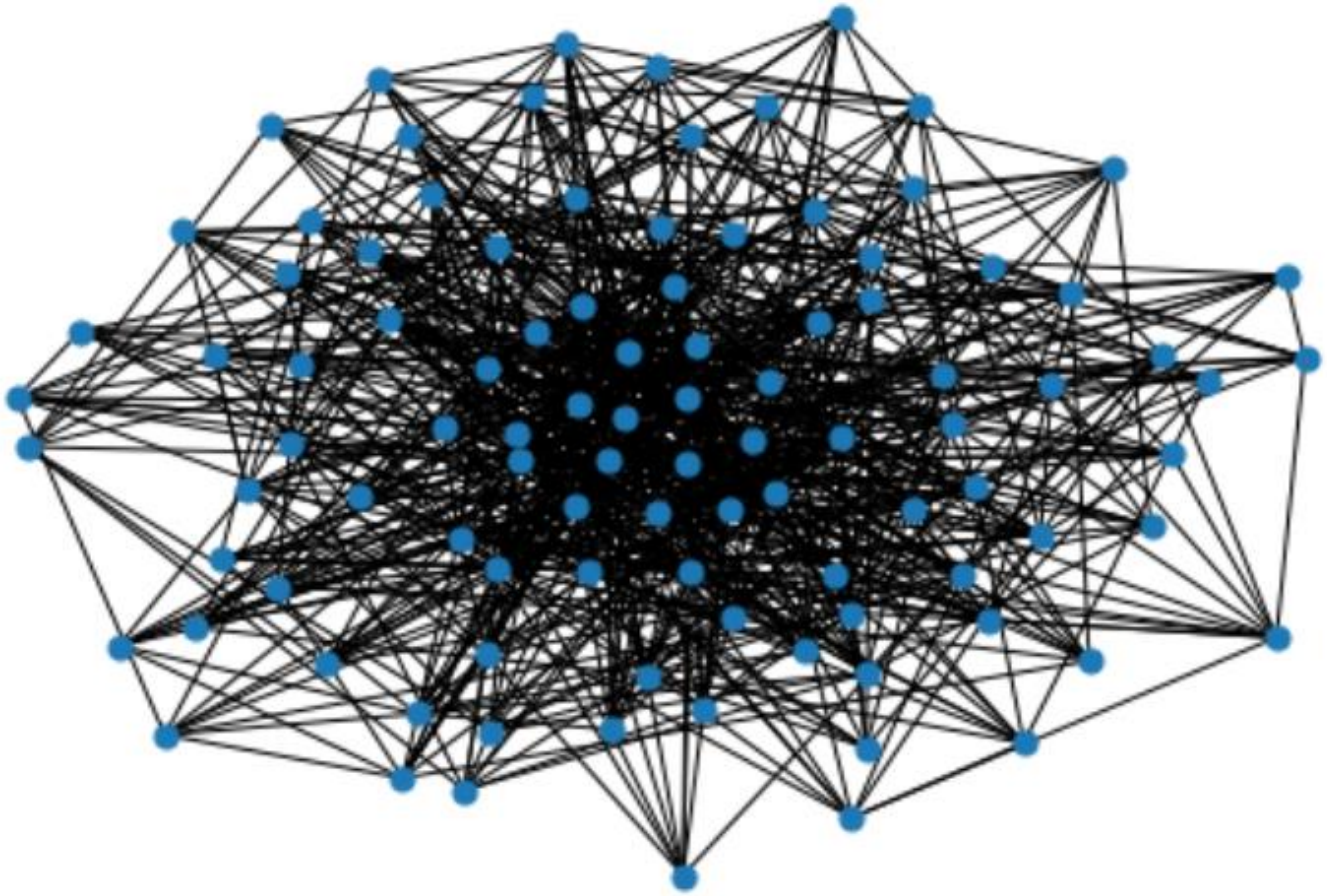
```python
plot_degree_distribution(G)
```



loglog scale!

Simplified Example Graph

barabasi_albert_net(10, 10, 100)



Works Cited

https://en.wikipedia.org/wiki/Barab%C3%A1si%E2%80%93Albert_model