Heidelberg University
Institute of Computer Science
Database Systems Research Group

Lecture: Complex Network Analysis

Prof. Dr. Michael Gertz

Assignment 5
Growth and Preferential Attachment

`https://github.com/nilskre/CNA_assignments`

Team Member:	Patrick Günther, 3660886, Applied Computer Science rh269@stud.uni-heidelberg.de
Team Member:	Felix Hausberger, 3661293, Applied Computer Science eb260@stud.uni-heidelberg.de
Team Member:	Nils Krehl, 3664130, Applied Computer Science pu268@stud.uni-heidelberg.de

1 Problem 5-1 Configuration Model

1. Such a network cannot exist. The first node has a degree of 4. In the remaining network (without this node), only 2 stubs exist (node 2: 1 and node 3:1). This means there are not enough stubs to connect the stubs of node 1 to, without creating self-loops.

2. The network can exist.

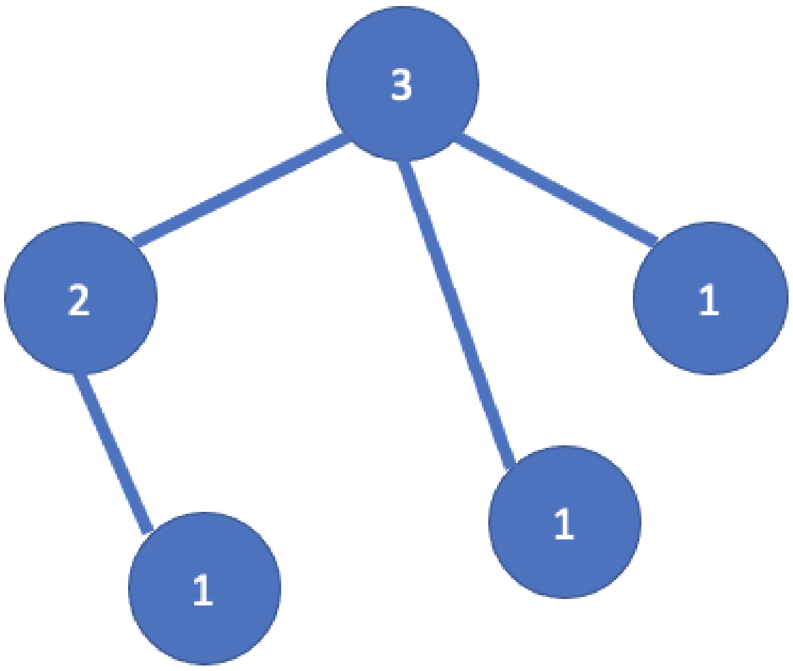


Figure 1: Example Network

3. Such a network cannot exits. If the connections for the first node with degree 3 are fulfilled, there are only two stubs remaining. Since this first node has to connect to all other nodes to obtain three connections, the remaining stubs can only be of the second node with degree 3. To achieve the degree of 3 for this node, the two remaining stubs would have to be connected to each other, creating a self-loop, which is not allowed.

2 Problem 5-2 Role of Preferential Attachment

To verify that preferential attachment is a necessary ingredient to build a model of a scale-free network (cf. Slide 5-19), Barabási and Albert also studied alternative versions of their algorithm. In the following, we want to explore how the omission of preferential attachment affects the degree distribution. Note that all tasks can be solved independently! If you skip a task, you may use the corresponding result without a proof. Assume a network is generated as follows: We start with m_0 initial nodes. Links between those nodes are chosen arbitrarily, as long as each node has a link. At each time step, a new node with m links connecting to existing nodes is added. However, in contrast to the model presented in the lecture, the probability that a link of a new node connects to node i does not depend on the degree k_i. We assume a uniform distribution, i.e., all nodes are equally likely and the probabilities are all equal to:

$$\prod = \frac{1}{m_0 + t - 1} \tag{1}$$

1. Show that each node i acquires links according to the following differential equation:

$$\frac{dk_i}{dt} \approx \frac{m}{m_0 + t - 1} \tag{2}$$

Equation (5.2) in Slide 5-8 describes the rate at which existing node i acquires links:

$$\frac{dk_i}{dt} = 1 - (1 - \prod(k_i))^m \approx m \prod(k_i) \tag{3}$$

For $\prod$ we insert equation 1 from above:

$$\frac{dk_i}{dt} = m\frac{1}{m_0 + t - 1} = \frac{m}{m_0 + t - 1} \tag{4}$$

---

2. Show that by separating variables and integrating, we obtain the following formula for $k_i(t)$:

$$k_i(t) = m\left[1 + log(\frac{m_0 + t - 1}{m_0 + t_i - 1})\right] \tag{5}$$

Hint: Note that node i joins the network at time t_i with m links, i.e., $k_i(t_i) = m$.

Start with equation 2:

$$\frac{dk_i}{dt} \approx \frac{m}{m_0 + t - 1}$$

Move dt to the other side by multiplying:

$$dk_i \approx \frac{m}{m_0 + t - 1} * dt$$

Integrate:

$$\int_m^{k_i(t)} \frac{1}{k_i}\, dk_i = \int_{t_i}^{t} \frac{m}{m_0 + t - 1}\, dt$$

Build Antiderivatives F ($f(x) = \frac{n}{x} => F(x) = n * log(x)$) and solve integral by $\int_a^b f(x)\, dx = F(b) - F(a)$:

$$log(k_i(t)) - log(m) = (m * log(m_0 + t - 1)) - (m * log(m_0 + t_i - 1))$$

Now extract m on the right side and use log calculation rules $log(u) - log(v) = \frac{log(u)}{log(v)}$

$$log(k_i(t)) - log(m) = m * (\frac{log(m_0 + t - 1)}{log(m_0 + t_i - 1)})$$

Move $log(m)$ to the other side by addition:

$$log(k_i(t)) = m * (\frac{log(m_0 + t - 1)}{log(m_0 + t_i - 1)}) + log(m)$$

Remove log from both sides:

$$k_i(t) = m * (\frac{log(m_0 + t - 1)}{log(m_0 + t_i - 1)}) + m$$

Reformulate:

$$k_i(t) = m * \left[1 + log(\frac{m_0 + t - 1}{m_0 + t_i - 1})\right]$$

---

3. Starting from the assertion $k_i(t) < k$, show that nodes have a degree smaller than k exactly if:

$$t_i > 1 - m_0 + (m_0 + t - 1)exp(1 - \frac{k}{m}) \tag{6}$$

Starting from $k_i(t) < k$, insert equation 5 for $k_i(t)$:

$$m * \left[1 + log(\frac{m_0 + t - 1}{m_0 + t_i - 1})\right] < k$$

Divide by m:

$$1 + log(\frac{m_0 + t - 1}{m_0 + t_i - 1}) < \frac{k}{m}$$

Substract $\frac{k}{m}$:

$$1 - \frac{k}{m} + log(\frac{m_0 + t - 1}{m_0 + t_i - 1}) < 0$$

Split log into two parts ($log(\frac{u}{v}) = log(u) - log(v)$):

$$1 - \frac{k}{m} + log(m_0 + t - 1) - log(m_0 + t_i - 1) < 0$$

Use exponential function ($k = log(c) <=> e^k = c$):

$$e^{1-\frac{k}{m}} + (m_0 + t - 1) - (m_0 + t_i - 1) < 0$$

Move t_i to the other side (first: $+(m_0 + t_i - 1)$ second: $+1 - m_0$):

$$e^{1-\frac{k}{m}} + (m_0 + t - 1) < m_0 + t_i - 1$$

$$1 - m_0 + (m_0 + t - 1) + e^{1-\frac{k}{m}} < t_i$$

Somewhere in our derivation there is still a fault left, because our final expression slightly differs from equation 6 (the exponential term is summed and not multiplied).

Problem 5-3 Barabasi-Albert Model

November 28, 2021

1 Lecture: Complex Network Analysis

Prof. Dr. Michael Gertz

Winter Semester 2021/22

1.1 Assignment 3 - Growth and Preferential Attachment

Students: Felix Hausberger, Nils Krehl, Patrick Günther

[1]:
```
import networkx as nx
import numpy as np
import powerlaw as pl
import matplotlib.pyplot as plt
```

1.2 1.

[2]:
```
def barabasi_albert(G, t, m):
    N_0 = G.number_of_nodes()
    for node in range(N_0, N_0 + t):
        G.add_node(node)
        N = G.number_of_nodes()
        links_added = 0
        while(links_added < m):
            link_probabilities = np.empty(N)
            sum_of_degrees = np.sum([G.degree(n) for n in G.nodes()])
            for source_node, degree in G.degree():
                link_probabilities[source_node] = degree / sum_of_degrees
            target_node = np.random.choice(N, p=link_probabilities)
            if(source_node != target_node and not G.has_edge(source_node,␣
 ↪target_node)):
                G.add_edge(source_node, target_node)
                links_added += 1;
    return G
```

1.3 2.

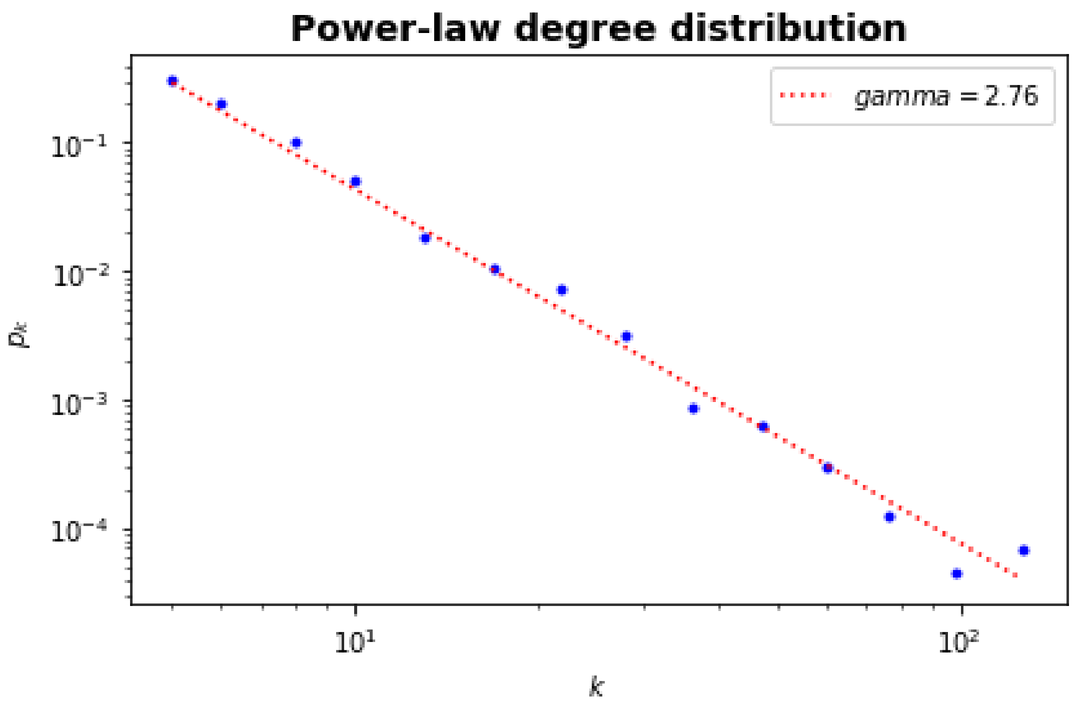
```
[3]: G = barabasi_albert(nx.complete_graph(5), 100, 3)
     print(f"Number of nodes: {G.number_of_nodes()}")
     print(f"Number of edges: {G.number_of_edges()}")
     print(f"Sum of the node degrees: {np.sum([G.degree(n) for n in G.nodes()])}")
```

```
Number of nodes: 105
Number of edges: 310
Sum of the node degrees: 620
```

1.4 3.

```
[4]: G = barabasi_albert(nx.complete_graph(5), 1000, 4)
     N = G.number_of_nodes()

     x, y = pl.pdf([G.degree(n) for n in G.nodes()], linear_bins=False)
     fit = pl.Fit([val for (node, val) in G.degree()], discrete=True)

     fig, ax = plt.subplots()
     ax.semilogx(x[1:], y, "b.")
     fit.power_law.plot_pdf(ax=ax, linestyle=":", color="r", label="$gamma = {}$".
      ↪format(np.round(fit.alpha, 2)))
     ax.set_title("Power-law degree distribution", fontweight="bold", fontsize=14)
     ax.set_ylabel("$p_k$")
     ax.set_xlabel("$k$")
     ax.legend()
     fig.tight_layout()

     print(f"Average local clustering coefficient: {np.round(nx.
      ↪average_clustering(G), 2)} (expected {np.round((np.log(N)**2)/N, 2)})")
     print(f"Average distance: {np.round(nx.average_shortest_path_length(G), 2)}␣
      ↪(expected {np.round(np.log(N)/np.log(np.log(N)), 2)})")
     print(f"Power-law degree exponent: {np.round(fit.alpha, 2)} (expected {3})")
```

```
Calculating best minimal value for power law fit
D:\Benutzer\Felix\anaconda3\lib\site-packages\powerlaw.py:699: RuntimeWarning:
invalid value encountered in true_divide
  (CDF_diff**2) /
D:\Benutzer\Felix\anaconda3\lib\site-packages\powerlaw.py:699: RuntimeWarning:
divide by zero encountered in true_divide
  (CDF_diff**2) /
D:\Benutzer\Felix\anaconda3\lib\site-packages\powerlaw.py:699: RuntimeWarning:
invalid value encountered in true_divide
  (CDF_diff**2) /

Average local clustering coefficient: 0.04 (expected 0.05)
Average distance: 3.19 (expected 3.58)
Power-law degree exponent: 2.76 (expected 3)
```



The values of the generated instance approach the expected values. The expected values are based on an analytical formula for the case that $t \to \infty$. As we only generate a small network, approaching the analytical with our experimental values is fine.